LA PRIMERA REVISTA DE PROGRAMACION EN CASTELLANO

SÓLO PROGRAMADORES

AÑO VI. 2.ª ÉPOCA NÚMERO 69 UNA PUBLICACIÓN DE: REVISTAS PROFESIONALES S.L. Precio: 1.250 Ptas. 7,5 € (IVA incluido)

NÚMERO EXTRA

HERRAMIENTAS:

- Optimize It contra JProbe
- Fastlib 2000 para Windows
- Voltímetro y osciloscopio virtual

LINUX:

- Configuración de X-Window en Linux

TELECOMUNICACIONES:

- WAP: El uso de scripts

VISUAL BASIC:

- Programación orientada a objetos

BASES DE DATOS:

- Bases de datos en Delphi

ENTORNOS DE DESARROLLO:

- Visual C++ y MFC

DELPHI:

- Programación OpenGL

XML / JAVA, tacada segura

CONTENIDO CD
Base de Datos SQL
INTERBASE 5.6
y 92 APLICACIONES Y LIBRERÍAS

8 413042 303299 00069

SUMARIO 69

SÓLO PROGRAMADORES

6 Noticias

Un repaso por la actualidad de la programación y la informática. En esta sección encontrarás las novedades que más nos han llamado la atención y las noticias más importantes del mes.

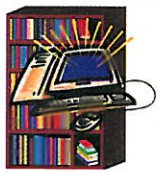
22 Telecomunicaciones

CURSO DE PROGRAMACIÓN CON WAP (IV)

Tras aprender en el número anterior a ampliar las prestaciones de los móviles *WAP* mediante el uso de las variables, en esta ocasión, y siguiendo con nuestro objetivo de alcanzar la mayor flexibilidad en la creación de contenido *WAP*, nos proponemos describir el uso de *Scripts* para *WML*. *WMLScript* es un lenguaje de procedimiento que está basado en *ECMAScript*, y es una de las capas de *WAP*, pudiendo ser usado para crear procedimiento del lado del cliente.

28 Herramientas

FASTLIB 2000 PARA WINDOWS



Alguien dijo que en el futuro, la programación sería un arte consistente en "amasar" un conjunto de componentes de diferentes fabricantes para hacerlos funcionar armónicamente. Esta completísima librería va a ayudarnos en ese arte.

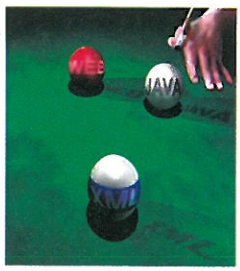
36 Visual Basic

PROGRAMACIÓN ORIENTADA A OBJETOS (y IV)

Acabamos la serie dedicada a la Programación Orientada a Objetos con dos ejemplos que mostrarán las posibilidades que ofrece la *POO* dentro de *Visual Basic*, con la implementación de dos clases. En primer lugar veremos una clase que utilizaremos para tratar con el acceso de datos de los objetos *ADO*, y en segundo lugar una clase que servirá para acceder a nuestros ficheros de inicio.

12 Portada Programación WEB

XML & JAVA, TACADA SEGURA



La tecnología *XML* es una de las que más recientemente se están desarrollando en todos los sitios de desarrollo de aplicaciones *Web*, y en la actualidad están surgiendo multitud de *API's* preparados para trabajar con *Java* y *XML*. El lenguaje *HTML* es una buena elección para presentar contenidos, pero no sirve para almacenar y representar datos. Sin embargo, los documentos *XML* representan un sistema sencillo y flexible que se adapta de forma natural a las tecnologías existentes. Por esto, *Java* resulta una opción excelente para utilizar *XML*.

42 Herramientas

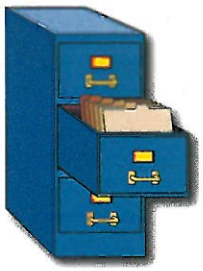
COMPARATIVA: OPTIMIZEIT CONTRA JPROBE

Existe en el mercado una amplia oferta de productos destinados a mejorar el rendimiento y la estabilidad de las aplicaciones en *Java*, eliminando los *loiterers* y alcanzando la máxima eficiencia en tareas como la creación de objetos. En este artículo vamos a presentar y comparar dos de los programas más utilizados para conseguir nuestro objetivo: *OptimizeIt*, de Intuitive Systems, y *JProbe*, de *KLF Group*. Nos encontramos ante dos productos de lo más completo, y ambos tienen características muy parecidas..., plataformas similares, información, *Java* 2, gestión del recolector de basura, control del *heap*, estadísticas, etcétera. ¿Por cuál decidirnos?

50 Bases de datos

APLICACIONES DE BASES DE DATOS CON DELPHI 5 (VI)

Continuamos con esta interesante serie, y tras ver en el artículo anterior la manera de conectar nuestras aplicaciones *Delphi* con cualquier sistema gestor de bases de datos, activando el servidor *Interbase/Oracle*, ahora estudiaremos la forma de crear mediante la versión 5 de *Delphi*, aplicaciones de bases de datos sin necesidad del *BDE*, usando los componentes *ADO* e *IB Express*. Esta característica resulta de gran utilidad para distribuir cualquier desarrollo que requiera bases de datos.



56 Linux

CONFIGURACIÓN DE X-WINDOW EN LINUX

Linux va perfeccionándose con el tiempo, y entre las mejoras que más van a agradecer sus usuarios encontramos a las herramientas gráficas de configuración. En este artículo veremos cómo funcionan, y su utilidad para la configuración del entorno *X-Window*, así como las variaciones encontradas durante el proceso de instalación de la nueva versión 4.0 de *XFree86*.

62. Entornos de 64 Desarrollo

VISUAL C++ y MFC (y VI)

La mayoría de las aplicaciones *Windows* necesitan en algún momento la acción concreta del usuario, ya sea introduciendo datos, o pulsando un botón concreto. Habitualmente, la petición de datos se produce con la visualización de ventanas, o cajas de diálogo. Terminamos esta serie analizando las características de las cajas o cuadros de diálogo de *Windows* y los controles más comunes, para alcanzar una completa interacción con nuestros programas.

70 Delphi

PROGRAMACIÓN OPENGL EN DELPHI (IV)

En el número anterior aprendimos a crear objetos complejos a partir de otros más simples. Siguiendo la serie, en esta entrega vamos a ver cómo se usan las primitivas relacionadas con colores, las luces y las sombras, de forma que den calidad a las escenas, acercándonos al realismo fotográfico. Entre las más utilizadas en informática para la representación cromática, la más común es la que utilice el espacio de color *RGB* Empecemos, pues, por repasar las diferentes combinaciones en la obtención de color.

78 Herramientas

ADQUISICIÓN DE DATOS CON C++ BUILDER. VOLTÍMETRO Y OSCILOSCOPIO VIRTUAL

Estamos seguros de que os va a gustar este artículo. Son cada vez menos los campos vedados a la informática... La adquisición de datos y la instrumentación virtual entran en la física, permitiendo procesar información eléctrica para traducirla en otras magnitudes.

94 Contenido del CD-ROM

Como de costumbre, en el *CD* encontraréis los códigos fuente y ejemplos de los artículos, además de un montón de aplicaciones y herramientas de gran utilidad; lenguajes, redes, entornos de programación, actualizaciones...

92 Libros

En estas páginas encontraréis una selección de las novedades editoriales dedicadas a la informática y la programación más interesantes. Como siempre, cada reseña va acompañada de una ficha con información de precio, idioma, editorial, etc.

96 Dudas Técnicas

Dedicamos este espacio a contestar vuestras preguntas y resolver cualquier duda que queráis plantear. No vaciléis en mandarnos vuestras consultas a la dirección **solop@revistasprofesionales.com** Como siempre, estaremos encantados de atenderos.

¡MODE, TRAS LOS PASOS DE WAP

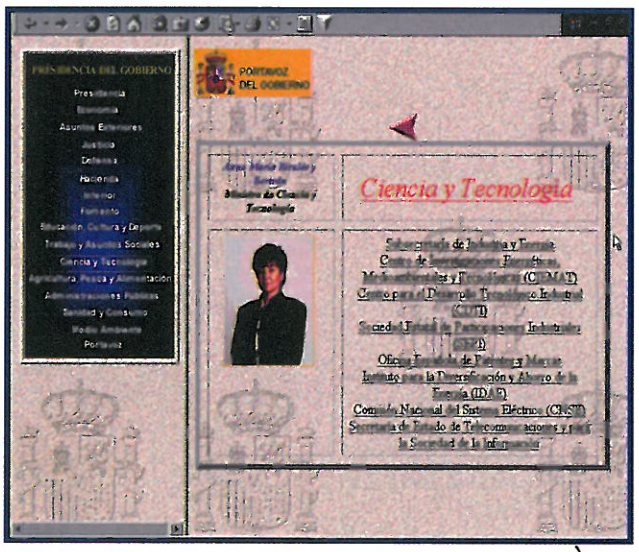
El sistema interactivo inalámbrico *iMode,* creado por el operador japonés *NTT DoCoMo*, está experimentando un notable desarrollo que amenaza la hegemonía de *Wap* en el sector de la difusión de contenidos de *Internet* por telefonía móvil.

Este servicio, basado en *HTML* y un paquete de protocolo inalámbrico, cuenta ya con 7 millones de abonados en Japón, tiene un sistema de conmutadores que proporciona mayor rapidez, versatilidad y eficiencia que la tecnología de circuitos empleada en Europa.

Mientras la implantación del servicio *iMode* en el Reino Unido avanza, *Wap* trabaja en el desarrollo de *GPRS* (servicio de datos inalámbricos de alta velocidad). Otra baza de *iMode* es la estabilidad de los asociados de *NTT DoCoMo*, aunque según *Michele Mackenzie*, analista de *Ovum*, "Existe un buen número de intereses detrás de *Wap*, por lo que será difícil verla caer".

LA TARIFA PLANA ANUNCIADA POR EL GOBIERNO NO CONVENCE A LA AUI

Miguel Pérez Subías, presidente de la *Asociación de Usuarios de Internet*, ha calificado el anuncio del Gobierno de la aplicación de tarifa plana en Internet, como una oferta de algo que ya existe en el mercado.



Pérez Subías señaló que ya existen bonos de 50 horas mensuales en horario reducido, al precio de 3.000 pesetas, siendo pocos los clientes que superan ese tiempo de conexión a la *Red*. De esta forma, la oferta del Gobierno, no supone ninguna aportación nueva que venga a solucionar a corto plazo el problema de las barreras existentes para el desarrollo de la competencia en el campo de la interconexión digital por red.

El presidente de la *AUI* reiteró la necesidad de un análisis serio de la situación que impide el desarrollo la tarifa plana, como paso previo a la propuesta de nuevos productos.

Recordemos que en la actualidad, la operadora de cable ONO está ofertando tarifa plana sin límite de horario por 2.500 Pts. al mes para Alicante y Castellón.

NUEVO MÓDEM DE ALTA SEGURIDAD

La compañía *Intel* ha firmado un acuerdo con *Network ICE Corporation*, por el que ésta incluirá en su *modem PRO/DSL* el *firewall* personal y de detección de intrusos de *Network ICE Corporation*. Este programa permite monitorizar el proceso de intercambio de información mediante *Internet* en líneas *ASDL*, de manera que se puedan identificar y bloquear accesos no autorizados o maliciosos al ordenador.

Este *modem* (aún en fase de pruebas), permitirá el acceso a los contenidos de *Internet* con una velocidad 150 veces superior a la proporcionada por los analógicos más rápidos disponibles hoy en el mercado.

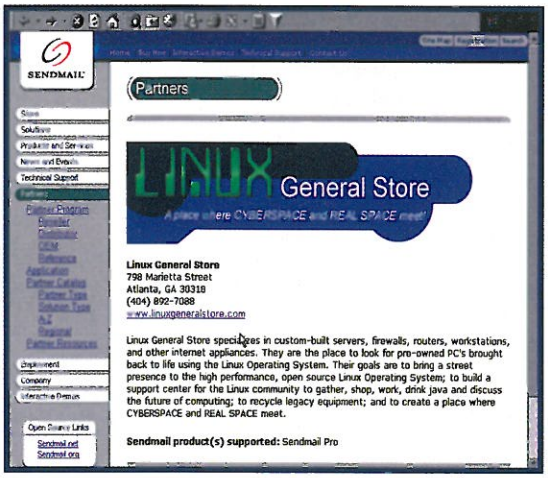
Otra característica de este dispositivo, es el acceso a la *Red* permanentemente abierto, y no es necesario hacer una llamada previa a la conexión con *Internet*.

LINUX, SEGURO PERO MENOS

La hasta ahora indiscutida seguridad del sistema operativo *Linux*, ha sufrido un contratiempo de consideración, al detectarse un virus en el núcleo del *SO* que afecta a un gran número de programas, especialmente el *Sendmail*.

Los problemas ocasionados por este virus en las versiones del núcleo *2.2.15* y *2.4.0* son de tal magnitud, que *Sendmail* aconseja actualizar estas versiones a la *2.2.16*, como garantía de que otros programas de este *SO* no se vean afectados.

A pesar de todo, en opinión de diversos expertos, la vulnerabilidad que supone el libre acceso al código fuente de *Linux*, se ve contrarrestada por la rapidez y eficiencia con la que se aportan soluciones, con lo que la seguridad de *Linux* está todavía por encima de la de *Windows NT*.

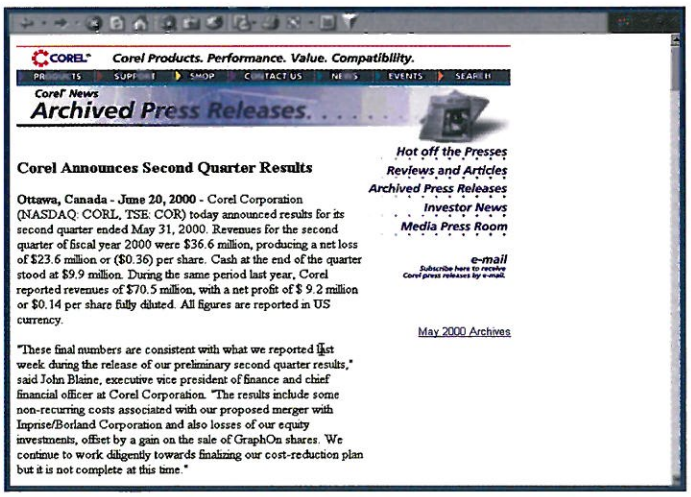

SENDMAIL
Partners
LINUX General Store
A place where CYBERSPACE and REAL SPACE meet
Linux General Store
798 Marietta Street
Atlanta, GA 30318
(404) 892-7088
www.linuxgeneralstore.com
Linux General Store specializes in custom-built servers, firewalls, routers, workstations, and other internet appliances. They are the place to look for pre-owned PC's brought back to life using the Linux Operating System. Their goals are to bring a street presence to the high performance, open source Linux Operating System; to build a support center for the Linux community to gather, shop, work, drink java and discuss the future of computing; to recycle legacy equipment; and to create a place where CYBERSPACE and REAL SPACE meet.
Sendmail product(s) supported: Sendmail Pro

ACUERDO INVESTRÓNICA Y ESWARE LINUX

La compañía *Investrónica* ha llegado a un acuerdo con *Esware Linux*, para comercializar como mayorista este sistema operativo destinado al entorno corporativo, la ofimática y el hogar.

Investrónica comercializará junto al sistema operativo, sus servicios de asistencia y consultoría, lo que supone una interesante oferta en el mercado para todos los distribuidores.

COREL, AL BORDE DE LA RUINA


COREL
Corel Products. Performance. Value. Compatibility.
Corel News
Archived Press Releases
Hot off the Presses
Reviews and Articles
Archived Press Releases
Investor News
Media Press Room
e-mail
Corel Announces Second Quarter Results
Ottawa, Canada - June 20, 2000 - Corel Corporation (NASDAQ: CORL; TSE: COR) today announced results for its second quarter ended May 31, 2000. Revenues for the second quarter of fiscal year 2000 were $36.6 million, producing a net loss of $23.6 million or ($0.36) per share. Cash at the end of the quarter stood at $9.9 million. During the same period last year, Corel reported revenues of $70.5 million, with a net profit of $ 9.2 million or $0.14 per share fully diluted. All figures are reported in US currency.
"These final numbers are consistent with what we reported last week during the release of our preliminary second quarter results," said John Blaine, executive vice president of finance and chief financial officer at Corel Corporation. "The results include some non-recurring costs associated with our proposed merger with Inprise/Borland Corporation and also losses of our equity investments, offset by a gain on the sale of GraphOn shares. We continue to work diligently towards finalizing our cost-reduction plan but it is not complete at this time."
May 2000 Archives

La firma canadiense *Corel*, ha comunicado que las perdidas correspondientes al segundo trimestre del año, podrían llegar a los 24 millones de dólares (4.200.000.000 Pts.) A esto se añaden los más de 12 millones $ perdidos en el primer trimestre, lo que sitúa a la compañía en una situación de difícil solución.

Entre las medidas anunciadas para absorber estas pérdidas, una de las más importantes será la drástica reducción de plantilla, aunque algunos analistas ya han advertido que esto no llegará a la mitad de los 40 millones de reducción de gastos que necesita *Corel* para afrontar la crisis.

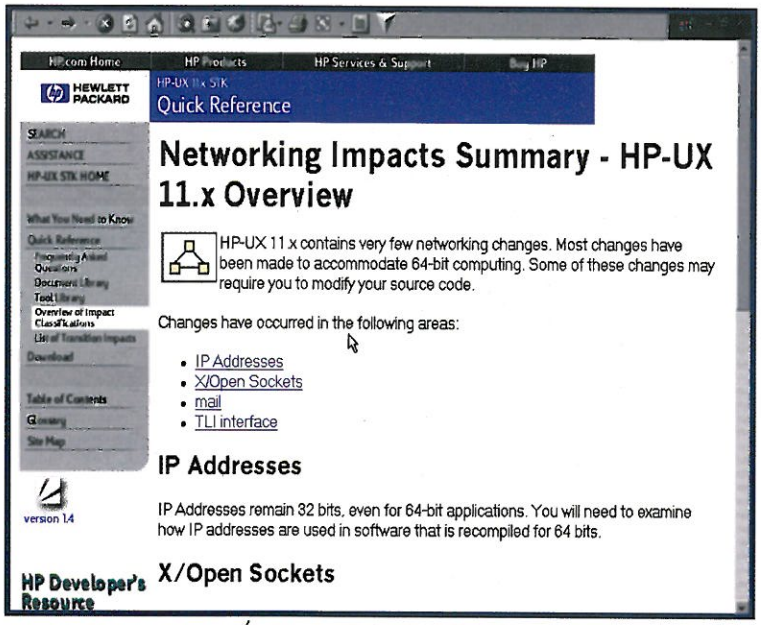
Recordemos que *Corel* había ingresado en el segundo trimestre más de 70 millones de dólares, pero la fallida operación de fusión con *Inprise/Borland*, situó a esta firma en una situación comprometida que ha vaciado sus arcas. Para *John Blaine*, director financiero de *Corel*, el primer objetivo de la empresa es el "realineamiento de la estructura de costes".

NUEVA VERSIÓN DEL SISTEMA OPERATIVO HP-UX

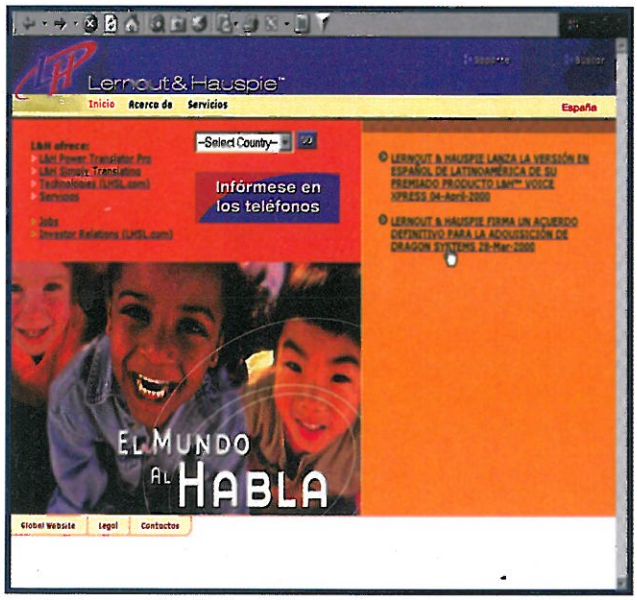
Hewlett-Packard ha presentado su último sistema operativo *Unix*, la última versión *HP-UX* de mayor capacidad y escalabilidad que sus antecesoras, y en la que se incluyen 25 nuevas características.

HP-UX 11X soporta 260 procesadores, 128 entornos *Hyperplex*, y cerca de 20.000 aplicaciones, además de aplicaciones *data-warehouse* e *Internet*, lo que en opinión de *Tony Iams*, analista de *DH Brown* coloca a *HP* muy por encima de otros sistemas operativos en el desarrollo de aplicaciones de alto nivel.

La última versión de *HP-UX*, cuenta ya con el respaldo y alianza estratégica de compañías como *Oracle, Informix, SAP, Baan* y *PeopleSoft*, según ha informado un portavoz de *HP*.


HP.com Home
HP Products
HP Services & Support
HEWLETT PACKARD
Quick Reference
Networking Impacts Summary - HP-UX 11.x Overview
HP-UX 11.x contains very few networking changes. Most changes have been made to accommodate 64-bit computing. Some of these changes may require you to modify your source code.
Changes have occurred in the following areas:
IP Addresses
X/Open Sockets
mail
TLI interface
IP Addresses
IP Addresses remain 32 bits, even for 64-bit applications. You will need to examine how IP addresses are used in software that is recompiled for 64 bits.
X/Open Sockets
version 1.4
HP Developer's Resource

NUEVA VERSIÓN DE POWER TRANSLATOR PRO

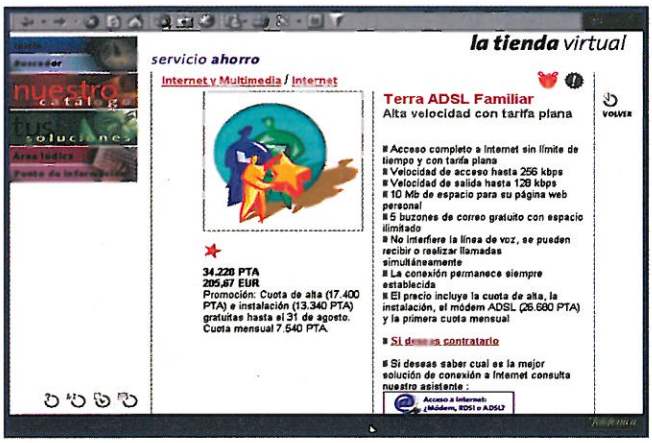


Lernout & Hauspie, compañía especializada en servicios y productos de reconocimiento de voz e idiomas, ha presentado la versión *7.0* del *Power Translator Pro*, el conocido programa de traducción de texto en diferentes documentos y *Web*.

Entre sus novedades, llaman la atención la renovada interfaz, los diccionarios ampliados, y la integración total con *MSOffice 2000*, además de la traducción bidireccional entre el español, el italiano y el inglés. Este mismo tipo de traducción también está disponible entre el inglés y el alemán, francés, portugués y japonés. Todas estas utilidades están disponibles para los procesadores de texto más comunes.

Otra interesante utilidad, es la ofrecida por la *Converstion Utility*, con la que dos interlocutores pueden comunicarse con idiomas diferentes en tiempo real mediante el teclado.

ADSL NO FUNCIONA


la tienda virtual
servicio ahorro
Internet y Multimedia / Internet
Terra ADSL Familiar
Alta velocidad con tarifa plana
VOLVER
34.228 PTA
205,67 EUR
Promoción: Cuota de alta (17.400 PTA) e instalación (13.340 PTA) gratuitas hasta el 31 de agosto. Cuota mensual 7.540 PTA.
Si deseas contratarlo

Durante los últimos días, Telefónica ha recibido multitud de quejas referidas al mal funcionamiento del servicio *ADSL* ofrecido por esta compañía a sus clientes.

En muchas ocasiones, el colapso del servicio ha sido total, lo que ha llevado a la indignación de particulares y empresas que dependen de este tipo de servicio para desarrollar su actividad.

Las explicaciones de telefónica no han podido ser más insatisfactorias para sus abonados, pues tras llamar al teléfono *902* de atención al cliente, sólo han obtenido un "acuse de recibo" de su queja, acompañado de algunas explicaciones técnicas más o menos complejas y poco convincentes, al tiempo que se les pedía más paciencia.

Telefónica empezó a ofrecer este servicio hace 7 meses, y a pesar de que sus deficiencias han sido graves y continuas, la compañía sigue facturando indefectiblemente a sus abonados por el uso de estas líneas, pensadas para alcanzar la máxima transmisión de datos por cable.

YA ESTÁ DISPONIBLE ADOBE LIVEMOTION

Adobe System Incorporated, una de las más importantes compañías en la publicación digital en *Web*, ha presentado su nueva herramienta *LiveMotion*, destinada a los diseñadores y desarrolladores de la *Web*.

Según *Adobe*, desde marzo se han descargado más de 350.000 versiones *beta* de este producto, lo que asegura la aceptación y demanda entre los diseñadores profesionales.

Una de las principales características de esta herramienta, es que trabaja con animaciones basadas en tiempo, y no en marcos, además de estar capacitada para operar con una gran gama de archivos.

Live Motion se presenta con el aval del premio obtenido en la feria *Spring Internet World 200*, celebrada recientemente en Los Ángeles.

CÓDIGO ABIERTO EN EL NUEVO WINDOWS CE PARA PORTÁTILES

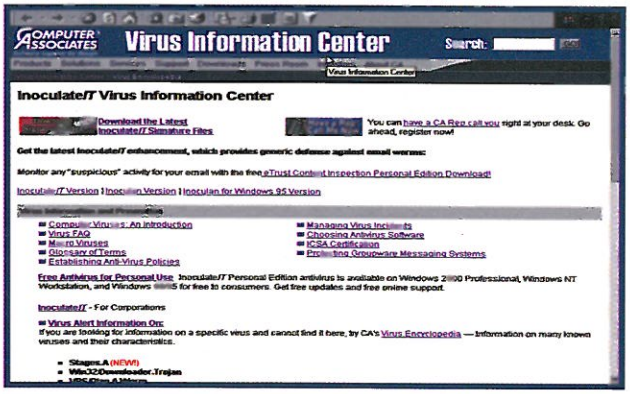
Microsoft hace una concesión a la creciente demanda de liberalización de códigos, y su nueva versión de *SO* para portátiles tiene gran parte del código fuente accesible, abriendo de esta forma la puerta al desarrollo de nuevas aplicaciones.

El nuevo *Windows CE*, ya está integrado en los equipos de bolsillo de *Compaq* y *HP*, y permite el acceso a *Internet*, reproducir música, además de soportar diversas aplicaciones gráficas, por lo que hace adecuada su implementación en otros tipos de dispositivos.

EL VIRUS SERBIAN BADMAN NO ES UNA AMENAZA INMEDIATA

La empresa *Computer Asociates Internacional* (*CA*) ha quitado importancia a las consecuencias de la aparición del troyano *Serbian Badman*, al notificar a sus clientes que este nuevo virus " no representa una amenaza inmediata crítica para los negocios electrónicos ni para los usuarios del hogar".

Serbian Badman es un típico "caballo de Troya" que no se replica ni infecta otros archivos, y se propaga por el correo electrónico y los grupos de información, o por descargas de archivos desde Internet, habiéndose detectado su presencia bajo la cobertura de una secuencia de película pornográfica disponible en la Red, desde la que intenta descargar el archivo *http://www.lomag.net/~ryan1918/MySissy.mpg.exe* e para ejecutarse tras su recibo.

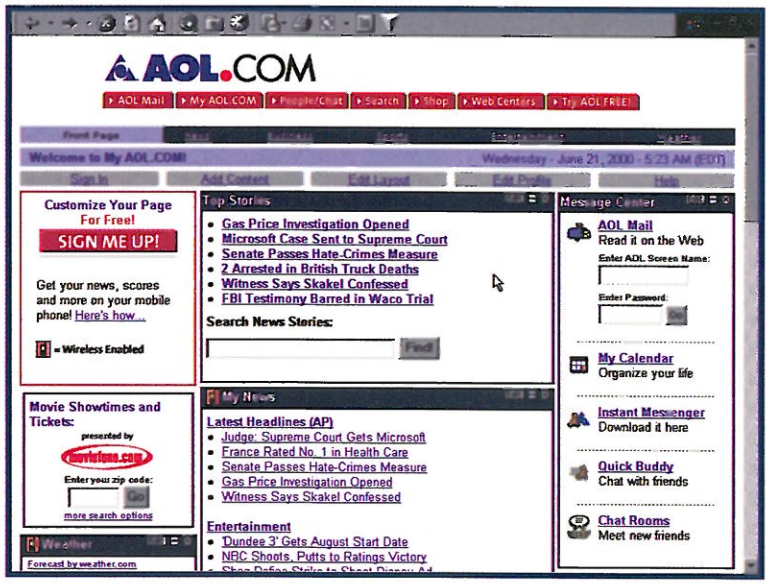

COMPUTER ASSOCIATES
Virus Information Center
Search:
Inoculate/T Virus Information Center
Download the Latest Inoculate/T Signature Files
You can have a CA Rep call you right at your desk. Go ahead, register now!
Get the latest Inoculate/T enhancement, which provides generic defense against email worms:
Monitor any "suspicious" activity for your email with the free eTrust Content Inspection Personal Edition Download!
Inoculate/T Version | Inoculan Version | Inoculan for Windows 95 Version
Computer Viruses: An introduction
Virus FAQ
Macro Viruses
Glossary of Terms
Establishing Anti-Virus Policies
Managing Virus Incidents
Choosing Antivirus Software
ICSA Certification
Protecting Groupware Messaging Systems
Free Antivirus for Personal Use: Inoculate/T Personal Edition antivirus is available on Windows 2000 Professional, Windows NT Workstation, and Windows 95/98 for free to consumers. Get free updates and free online support.
Inoculate/T - For Corporations
Virus Alert Information On:
If you are looking for information on a specific virus and cannot find it here, try CA's Virus Encyclopedia — information on many known viruses and their characteristics.
Stages.A (NEW!)
Win32/Downloader.Trojan

Los interesados en obtener información de este troyano, pueden dirigirse a la dirección http://www.ca.com/virusinfo/.

ATAQUE HACKER EN AMERICA ON LINE

America Online ha denunciado la invasión de las cuentas de sus empleados por un virus troyano. Este programa utilizó el correo electrónico para introducirse en la red de la compañía, y eludió las medidas de seguridad de *AOL* sin ser detectado por las barreras del sistema. Una vez instalado el caballo de Troya, quitó la protección de todos los datos confidenciales (información personal y económica) de los empleados de la compañía, de forma que estos quedaron a merced de los piratas informáticos.

No disponemos de datos sobre cómo fue detectado, ni de los daños que hasta el momento puede haber causado, aunque *AOL* ha manifestado que se están aplicando las medidas oportunas para solucionar el problema.



DISCO DURO PARA PORTÁTILES

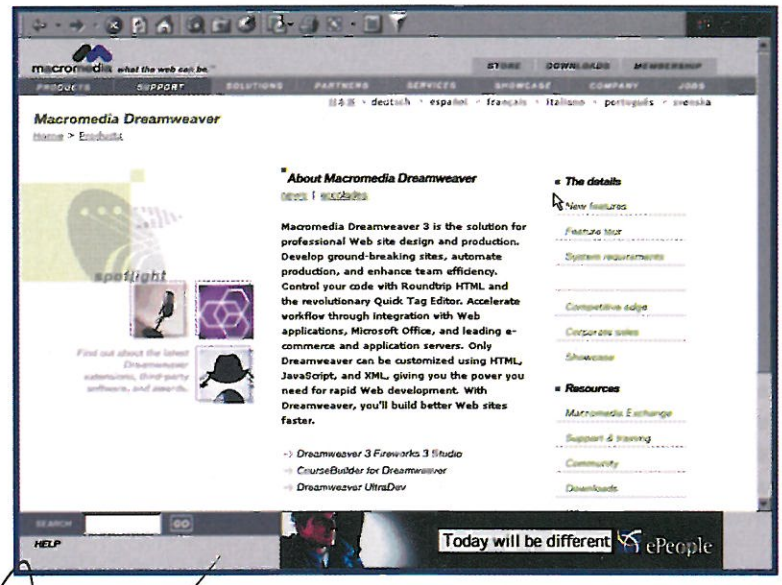
La compañía *IBM* trabaja en la fabricación de un disco duro de reducidas dimensiones, el *Microdrive,* con una capacidad de 1 *GB,* y destinado a ordenadores portátiles, agendas y cámaras digitales. Este dispositivo vendrá a aumentar notablemente la capacidad y prestaciones de los actuales equipos portátiles.

El *Microdrive* está todavía en fase de estudio, pero se prevé que su tamaño sea similar al de una memoria *Compact Flash II.*

El precio estimado de este dispositivo se acerca a las 90.000 pesetas.

VERSIÓN BETA DE WML

Macromedia y *Nokia* han presentado una versión *beta* de *WML Studio* de *Nokia* destinada al editor de páginas *Web Dreamweaver*. *WML Studio* permite el desarrollo de contenidos *WML* para su distribución entre los clientes de telefonía *Nokia*, facilitando la labor de los desarrolladores con funciones como la presentación preliminar de lo que el cliente va a ver en pantalla, y permitiéndoles crear tarjetas *WML*, modificar textos e insertar imágenes. Una de las principales ventajas de *WML* Studio, consiste en la creación de contenidos con el uso de objetos "arrastrar y soltar".


Macromedia Dreamweaver
About Macromedia Dreamweaver
Macromedia Dreamweaver 3 is the solution for professional Web site design and production. Develop ground-breaking sites, automate production, and enhance team efficiency. Control your code with Roundtrip HTML and the revolutionary Quick Tag Editor. Accelerate workflow through integration with Web applications, Microsoft Office, and leading e-commerce and application servers. Only Dreamweaver can be customized using HTML, JavaScript, and XML, giving you the power you need for rapid Web development. With Dreamweaver, you'll build better Web sites faster.
Today will be different
ePeople

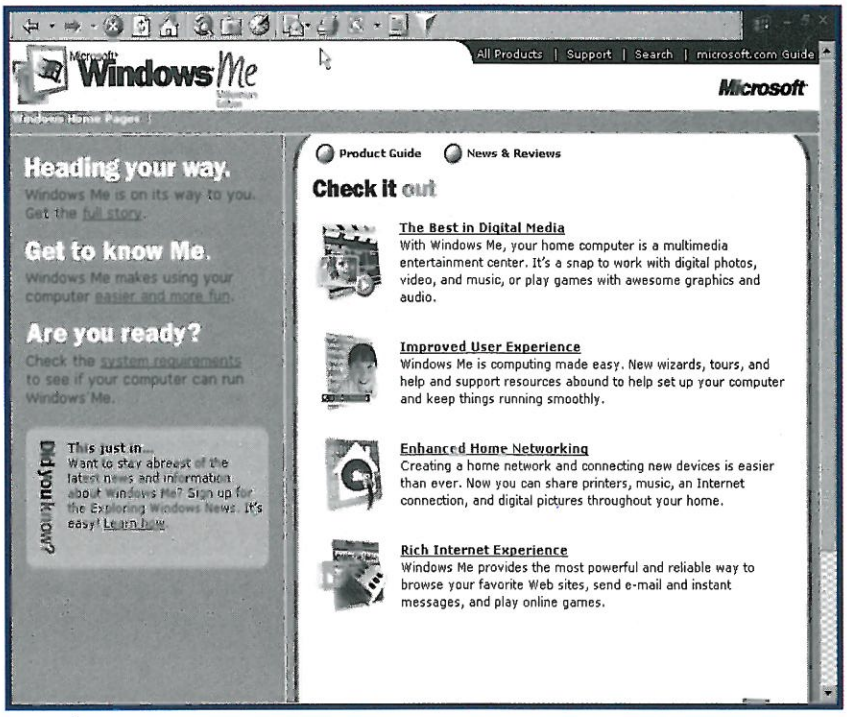
Esta versión beta, está a disposición de los usuarios en la dirección *Internet www.exchange.macromedia.com.*

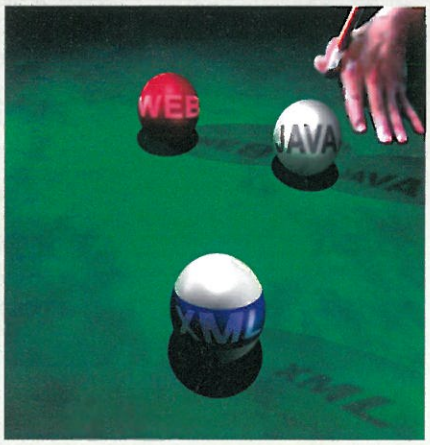
WINDOWS MILLENIUM SALE EN SEPTIEMBRE

Tras dar por terminadas las últimas pruebas, *Microsoft* ha empezado a repartir copias entre los distribuidores de *Windows Millennium*, la última versión del sistema operativo más popular de los últimos tiempos, y sucesor directo del *98 IIE*, la edición de la familia *Windows* destinada al hogar y la empresa.

Sin embargo, habrá que esperar hasta septiembre para poder encontrar el *ME (Millennium Edition)* en las tiendas de Estados Unidos, y algo más para el resto del mundo. La razón del retraso, obedece al margen de tiempo que los fabricantes de *hardware* necesitan para adaptar perfectamente el *SO* a sus equipos.

Se esperan importantes mejoras sobre anteriores versiones, sobre todo en el perfeccionamiento de herramientas y utilidades, aunque esta mejora irá acompañada de un mayor requerimiento de *hardware*. Esta circunstancia, a la que ya están acostumbrados y resignados los usuarios será bien recibida sin duda por los fabricantes de equipo, pues se prevé un aumento importante de venta de equipos nuevos.


Windows Me
Microsoft
Heading your way.
Get to know Me.
Are you ready?
Product Guide
News & Reviews
Check it out
The Best in Digital Media
With Windows Me, your home computer is a multimedia entertainment center. It's a snap to work with digital photos, video, and music, or play games with awesome graphics and audio.
Improved User Experience
Windows Me is computing made easy. New wizards, tours, and help and support resources abound to help set up your computer and keep things running smoothly.
Enhanced Home Networking
Creating a home network and connecting new devices is easier than ever. Now you can share printers, music, an Internet connection, and digital pictures throughout your home.
Rich Internet Experience
Windows Me provides the most powerful and reliable way to browse your favorite Web sites, send e-mail and instant messages, and play online games.



XML & JAVA (I), tacada segura

Adolfo Aladro García. *Analista Programador*

Las tecnologías *XML* y *Java* están condenadas a entenderse. La primera ofrece un medio portátil de tratamiento de datos a través de redes, mientras que *Java* es una plataforma nacida con *Internet*, y que ha cobrado gran importancia en este ámbito. Por tanto, ambas forman un tandem con un potencial enorme para el desarrollo de aplicaciones *Web*.

JAVA Y LA TECNOLOGÍA XML

En la actualidad, la plataforma *Java* es probablemente la única que proporciona los fundamentos para desarrollar y distribuir aplicaciones a través de las redes de manera segura y portátil. Ya no hablamos solamente de *applets* que se ejecutan en los navegadores de los usuarios, si no de aplicaciones *Web* complejas, donde están involucrados componentes de servidor (*servlets, Java Server Pages, Enterprise JavaBeans*, etcétera), acceso a base de datos (*JDBC*), comunicación con otros recursos internos o externos (*RMI, CORBA*, etcétera.)

La tecnología *XML* puede hacer lo mismo con los datos, ofreciendo un medio neutral o multiplataforma para representar los contenidos. Al fin y al cabo, los documentos *XML* no son más que ficheros de texto, como veremos más adelante.

XML puede modelar cualquier fuente de datos

La versión *1.0* del estándar *XML* fue publicada por el *W3C* (*World Wide Web Consortium*) el 10 de febrero de 1998. Desde entonces, la tecnología *XML* ha ido configurándose como un formato universal de intercambio de datos entre distintos tipos de sistemas. Entre sus ventajas podemos destacar:

- Puede modelar cualquier origen de datos, sea cual sea su complejidad estructural.
- Es extensible, en el sentido de que se pueden definir nuevas etiquetas siempre que sea necesario.
- Cuando los datos están en formato *XML* se puede comprobar fácilmente su validez estructural.
- *XML* es un estándar multiplataforma que no depende de sistemas operativos, bases de datos, etcétera.

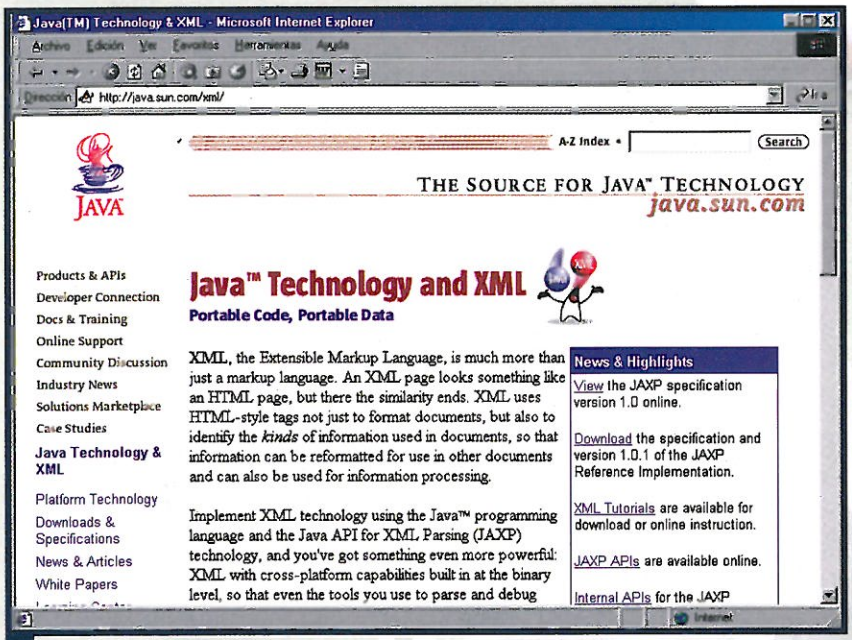

Java(TM) Technology & XML - Microsoft Internet Explorer
http://java.sun.com/xml/
A-Z Index
Search
THE SOURCE FOR JAVA™ TECHNOLOGY
java.sun.com
Products & APIs
Developer Connection
Docs & Training
Online Support
Community Discussion
Industry News
Solutions Marketplace
Case Studies
Java Technology & XML
Platform Technology
Downloads & Specifications
News & Articles
White Papers
Java™ Technology and XML
Portable Code, Portable Data
XML, the Extensible Markup Language, is much more than just a markup language. An XML page looks something like an HTML page, but there the similarity ends. XML uses HTML-style tags not just to format documents, but also to identify the kinds of information used in documents, so that information can be reformatted for use in other documents and can also be used for information processing.
Implement XML technology using the Java™ programming language and the Java API for XML Parsing (JAXP) technology, and you've got something even more powerful: XML with cross-platform capabilities built in at the binary level, so that even the tools you use to parse and debug
News & Highlights
View the JAXP specification version 1.0 online.
Download the specification and version 1.0.1 of the JAXP Reference Implementation.
XML Tutorials are available for download or online instruction.
JAXP APIs are available online.
Internal APIs for the JAXP

Figura 1- Sitio *Web* de *Sun* dedicado a las tecnologías *Java* y *XML* (*java.sun.com/xml/*).

Sun se ha decidido a dar soporte a la tecnología *XML*, definiendo una extensión *XML* estándar que se integra en el ámbito de la plataforma *Java*. Con dicha extensión, es posible leer, manipular y generar documentos *XML*. La extensión está basada en la especificación *1.0* del estándar *XML*, lo que incluye la recomendación del *W3C* (*W3C DOM Level 1 Core Recommendation*) y el *API SAX 1.0*.

XML va a revolucionar el desarrollo de aplicaciones Web

La tecnología *XML* va a revolucionar el desarrollo de las aplicaciones *Web*, especialmente aquellas en las que el intercambio de datos juega un papel indispensable. *Java*, como lenguaje de programación, proporciona recursos muy potentes para manejar cadenas de texto, tablas asociativas, acceso a *Internet* y otras muchas características que hacen que su colaboración con *XML* sea algo natural. *XML*, por su parte, contribuye a que *Java* cuente con un sistema de intercambio de datos portátil y sencillo.

XML FRENTE A HTML

Indudablemente, el lenguaje *XML* está muy unido al lenguaje *HTML* y esto puede generar bastante confusión. Lo cierto, es que cada vez surgen más y más siglas: *XHTML, XSL, XQL*, etcétera, y esta situación no contribuye precisamente a aclarar el asunto. A continuación, vamos a exponer algunos de los puntos principales que marcan la diferencia entre ambos lenguajes.

- El lenguaje *HTML* no es extensible. Esto significa que los desarrolladores no pueden crear sus propias etiquetas para resolver los problemas específicos de sus aplicaciones de una manera más efectiva.

- El lenguaje *HTML* está enfocado principalmente hacia la presentación de los contenidos. Las etiquetas *HTML* sirven para indicar cómo ha de mostrarse la información en el navegador, pero en cambio, no son útiles en aplicaciones de tratamiento de datos a través de la *Red*.

HTML no permite definir etiquetas personalizadas

- El lenguaje *HTML* no sirve para separar los contenidos de la presentación. Con frecuencia, encontramos páginas *Web* en *Internet* donde los contenidos varían dinámicamente. Las aplicaciones de servidor y tecnologías como las páginas *ASP*, las páginas *JSP*, *ColdFusion* o *Perl*, han conseguido resolver este problema generando las páginas *HTML* dinámicamente, pero lo anterior sólo refleja las carencias del lenguaje *HTML* para discernir entre los contenidos de una página y la manera en la que éstos se muestran.

- El lenguaje *HTML* no sirve para ofrecer distintas "vistas" de los datos. La información contenida en una página *Web* normalmente se muestra de una sola forma. Si bien esto puede modificarse con *HTML Dinámico* en el cliente, o con más llamadas al servidor, en realidad lo que se necesita es contar con la posibilidad de que el navegador reciba, por un lado, la información, y por otro, el código que proporciona el abanico de formas distintas de visualizar dicha información.

- El lenguaje *HTML* no tiene una estructura semántica. Uno de los problemas más importantes con los que cuentan los buscadores en *Internet*, es que resulta prácticamente imposible extraer cualquier

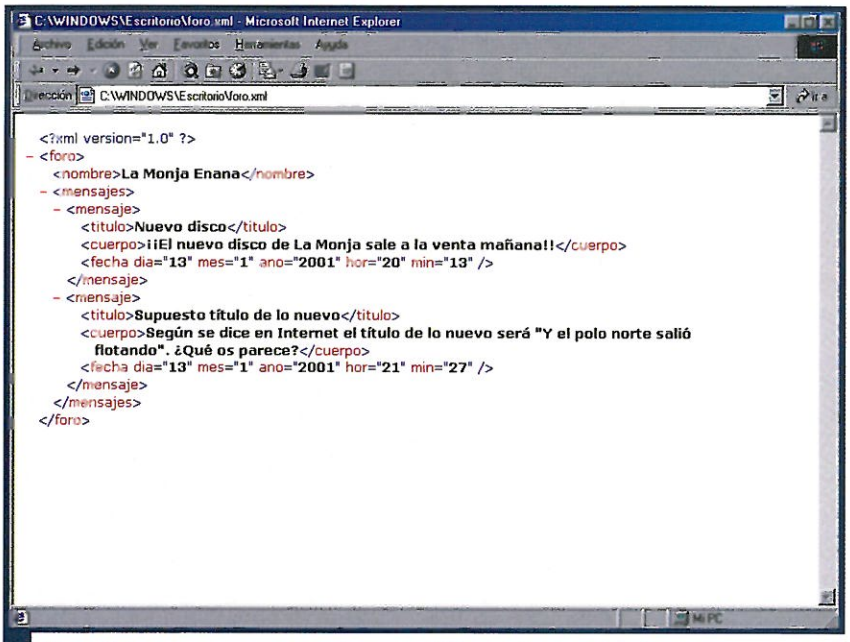


Figura 2- Aspecto que presenta un documento *XML* cuando se visualiza directamente con *Internet Explorer 5.x* o superior.

tipo de información referida al contenido de una página *Web*. Las etiquetas *HTML* sólo determinan el aspecto que tendrá la información.

FUENTES DE DATOS XML

Un fichero con extensión *XML*, no es más que un archivo de texto plano que cuenta con una estructura determinada. Por ejemplo:

```
Width1Width3Width8505 <?xml ver-
     sion='1.0'?>
<persona id="3462312">
  <nombre>Adolfo</nombre>
  <apellidos>Aladro</apellidos>

    <email>aaladro@arrakis.es</em
    ail>
</persona>
Width1Width3Width8505
```

La primera línea, es una directriz que indica que lo que viene a continuación es un documento *XML*, el cual se ajusta a la versión *1.0* del estándar.

Las etiquetas *HTML* sólo sirven para mostrar la información

Debemos tener en cuenta que toda fuente de datos *XML* tiene una estructura arborescente. Se parte de una raíz principal, a partir de la cual surgen distintos nodos, en tantos niveles como sea preciso para representar la información. En el ejemplo que nos ocupa, el nodo raíz es el definido por la etiqueta *<persona>*. De él surgen tres nodos más, que además, son finales, y que contienen en última instancia la información. Están representados por las etiquetas *<nombre>*, *<apellidos>* y *<email>*.

Como se puede observar, los documentos *XML* son muy sencillos de elaborar y se pueden adaptar con facilidad a cualquier caso en concreto. No sólo almacenan la información, sino que además indican cuál es su estructura.

DOCUMENTOS XML BIEN FORMADOS

Un documento *XML* bien formado, es aquel que respeta las reglas de estructura y notación. Las aplicaciones que trabajan con fuentes de datos *XML* tienen que comprobar en primer lugar la consistencia sintáctica de los documentos.

Un documento XML es un fichero de texto plano

El lenguaje *HTML* no es más que un subconjunto específico de *XML*. La mayor parte de los desarrolladores de *Internet* están familiarizados con él en un grado u otro. Por este motivo, la mejor manera de entender los factores que determinan que un documento *XML* esté bien formado, es analizar la sintaxis utilizada en las páginas *HTML* y compararla con las reglas de notación y estructura del lenguaje *XML:*

- No puede haber etiquetas sin cerrar.

En la páginas *HTML* se dan con frecuencia casos como el que sigue:

```
<P>Esto es un párrafo...<P>Y esto
    otro...
```

La etiqueta *<P>* tiene su correspondiente etiqueta de cierre, *</P>*, pero los navegadores son lo suficientemente flexibles como para revisar el código anterior e interpretarlo sin problemas. Sin embargo, ese código no es correcto

como documento *XML*, ya que existen etiquetas sin cerrar. Para corregirlo deberíamos hacer:

```
<P>Esto es un párrafo...</P><P>Y
    esto otro...</P>
```

o bien utilizar la nomenclatura "especial" para etiquetas que no cuentan con la etiqueta de cierre:

```
<P/>Esto es un párrafo...<P/>Y
    esto otro...
```

Cada etiqueta XML debe tener su correspondiente cierre

- El orden de apertura y cierre de las etiquetas debe ser correcto.

Otro de los casos que se repite usualmente es el que presenta el ejemplo siguiente:

```
<FONT
    FACE="Arial"><B>SPP</FONT></B
    >
```

Una vez más, tenemos que decir que los navegadores son perfectamente capaces de interpretar lo anterior, pero ese código no está bien formado desde el punto de vista de la tecnología *XML*. Lo correcto sería hacer:

```
<FONT
    FACE="Arial"><B>SPP</B></FONT
    >
```

o bien:

```
<B><FONT
    FACE="Arial">SPP</FONT></B>
```

- Hay que utilizar dobles comillas para los atributos.

Incluir una imagen dentro de una página *Web*, es tan fácil como hacer:

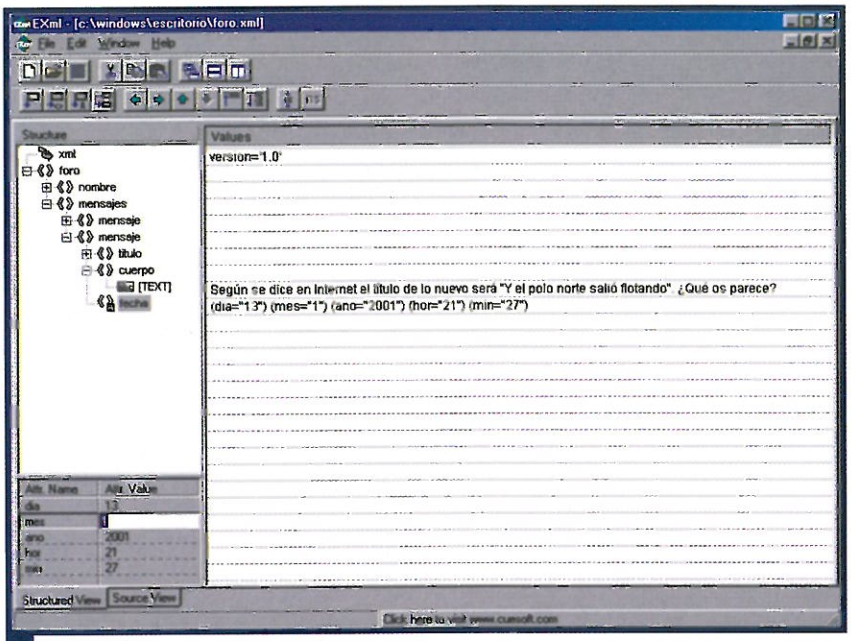


Figura 3- *EXml (www.cuesoft.com)* es uno de los editores de *XML freeware* más útiles y sencillos de manejar.

```
<IMG SRC=icono.gif WIDTH=120
    HEIGHT=30>
```

La utilización de esta etiqueta dentro de un documento *XML*, tal y como está expuesta, daría lugar a que éste no estuviera bien formado por dos motivos. El primero de ellos, es que no tiene etiqueta de cierre correspondiente, tal y como hemos indicado en un punto anterior. La solución sería hacer:

```
<IMG SRC=icono.gif WIDTH=120
    HEIGHT=30/>
```

Pero es que lo anterior tampoco sería válido, ya que todos los atributos están sin comillas. Por lo tanto la expresión correcta sería:

```
<IMG SRC="icono.gif" WIDTH="120"
    HEIGHT="30"/>
```

- Los caracteres especiales deben representarse mediante entidades especiales.

Todos los caracteres utilizados en un documento *XML* tienen su correspondiente entidad. Esto significa que por ejemplo, el carácter *a* puede representarse también con la cadena *a*, que no es más que lo que se denomina como "entidad de carácter". Cuando se utilizan caracteres especiales, como por ejemplo las vocales acentuadas, la utilización de entidades de carácter es obligatoria para que el documento *XML* esté bien formado.

Un documento bien formado ofrece muchas más garantías, y además, los *parsers* pueden procesarlo a mayor velocidad.

DOCUMENTOS XML VÁLIDOS

La validez de un documento *XML* representa un paso más allá de lo que hemos visto hasta el momento. Cuando se dice que un documento *XML* está bien formado, solamente significa que su sintaxis se ajusta a las reglas anteriormente expuestas. Esto no implica

necesariamente que los datos representados por dicho documento tengan sentido alguno.

Los atributos deben ir entrecomillados

Partiendo del ejemplo anterior, podemos definir una agenda personal en formato *XML* de la siguiente forma:

```
Width1Width3Width8446 <?xml ver-
    sion='1.0'?>
<agenda>
  <persona id="3462312">
    <nombre>Adolfo</nombre>
    <apellidos>Aladro</apellidos>

    <email>aaladro@arrakis.es</em
    ail>
  </persona>
  <persona>
    <nombre>Madonna</nombre>
    <apellidos>Ciccone</apelli-
    dos>
  </persona>
```

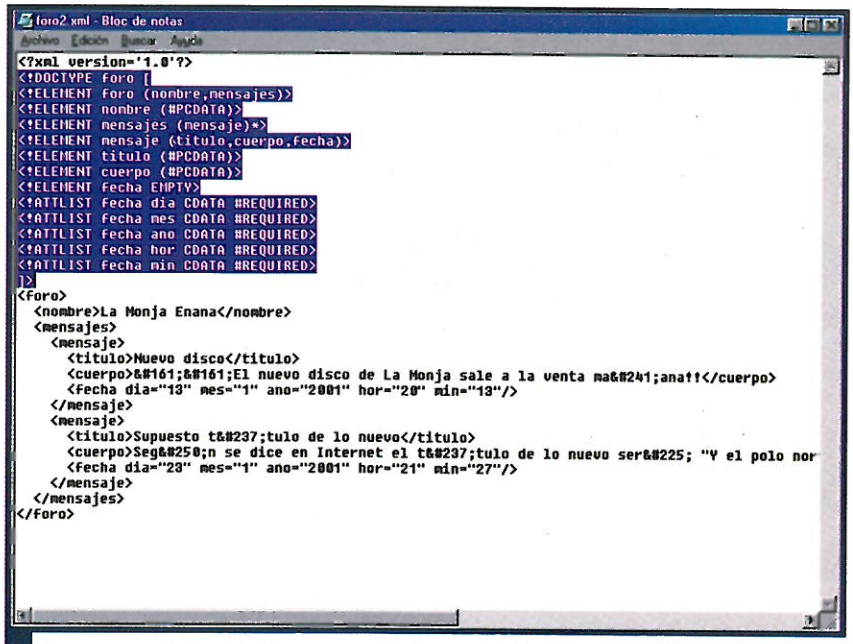


Figura 4- Los *DTD* pueden integrarse en el mismo archivo del documento *XML*.

Lo anterior representa un documento *XML* bien formado, ya que se respetan todas las reglas que hemos expuesto. Pero no debería ser válido, ya que existen dos elementos *persona* definidos de manera diferente, lo que genera una inconsistencia semántica.

Un documento XML tiene estructura arborescente

Supongamos ahora que vamos a utilizar la tecnología *XML* para representar foros de debate como los que pueblan numerosos sitios *Web* y portales de *Internet*. En la Tabla 1 podemos ver una primera aproximación a lo dicho.

El documento anterior expresa lo siguiente:

- Un foro tiene un nombre y un buzón con cero o más mensajes.
- Cada mensaje cuenta con un título, un cuerpo y la fecha en la que se creó.

Comprobar la validez de este documento *XML*, no sería más que verificar que se cumplen los dos puntos anteriores. Se denomina

TABLA 1. Representación de un foro de debate

```
<?xml version='1.0'?>
<foro>
 <nombre>La Monja Enana</nombre>
 <mensajes>
  <mensaje>
   <titulo>Nuevo disco</titulo>
   <cuerpo>&#161;&#161;El nuevo disco de La Monja
       sale a la venta ma&#241;ana!!</cuerpo>
   <fecha dia="13" mes="1" ano="2001" hor="20" min="13"/>
  </mensaje>
  <mensaje>
   <titulo>Supuesto t&#237;tulo de lo nuevo</titulo>
   <cuerpo>Seg&#250;n se dice en Internet el t&#237;tulo de lo
     nuevo ser&#225; "Y el polo norte sali&#243;
     flotando".&#13;&#10;&#13;&#10;&#191;Qu&#233;
     os parece?</cuerpo>
   <fecha dia="13" mes="1" ano="2001" hor="21" min="27"/>
  </mensaje>
 </mensajes>
</foro>
Width1Width3Width8446
```

DTD (Document Type Declaration) al documento que recoge precisamente las normas que dicen cuál es la disposición correcta de etiquetas y atributos para un origen de datos *XML* en concreto.

Los DTD se usan para comprobar la validez de los documentos XML

Los documentos *DTD* pueden ser archivos separados o integrados dentro del propio documento *XML*. También puede darse el caso de que no exista un documento *DTD* para un origen de datos *XML*. En ese caso se podría seguir comprobando si el documento *XML* está bien formado o no, pero no se podría decir nada de su validez.

ELABORAR UN DTD

Los documentos *DTD* son tan sólo ficheros de texto plano, en los que se define la estructura correcta de los datos *XML*. Para ello se utilizan una serie de etiquetas. A continuación, vamos a estudiar cómo se puede elaborar un documento *DTD* a partir del que hemos escrito para la fuente de datos correspondiente a un foro de debate.

```
<!ELEMENT foro (nombre,mensajes)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT mensajes (mensaje)*>
<!ELEMENT mensaje (titulo,cuer-
    po,fecha)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT cuerpo (#PCDATA)>
<!ELEMENT fecha EMPTY>
<!ATTLIST fecha dia CDATA #REQUI-
    RED>
<!ATTLIST fecha mes CDATA #REQUI-
    RED>
<!ATTLIST fecha ano CDATA #REQUI-
    RED>
<!ATTLIST fecha hor CDATA #REQUI-
    RED>
<!ATTLIST fecha min CDATA #REQUI-
    RED>
```

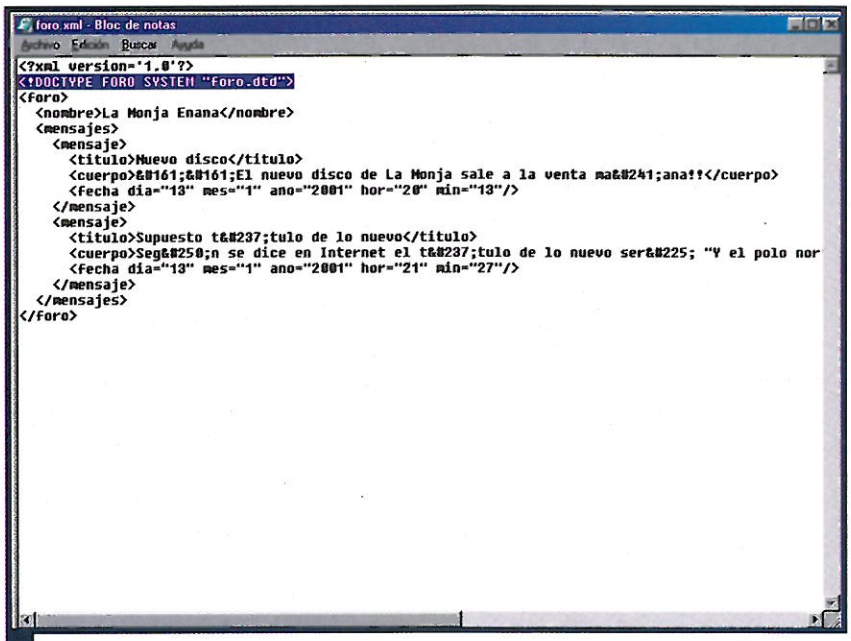


Figura 5- Los ficheros con extensión *DTD* se utilizar para validar fuentes de datos *XML*.

Los DTD pueden estar incluidos en el propio fichero XML

Nótese que cuando el documento *DTD* se incluye dentro del propio archivo *XML*, se utiliza la siguiente sintaxis:

```
<?xml version='1.0'?>
<!DOCTYPE foro [
    ...Declaraciones
]>
...
```

Si el documento *DTD* permanece como un archivo aparte entonces se hace:

```
<?xml version='1.0'?>
<!DOCTYPE foro SYSTEM "foro.dtd">
...
```

ELEMENTOS

En primer lugar encontramos etiquetas *ELEMENT* que permiten definir cómo son los elementos del origen de datos *XML*. La sintaxis de esta etiqueta es la mostrada en la Tabla 2.

El primer caso significa que el elemento *foro* está compuesto por otros los elementos *nombre* y *mensaje*. La segunda utilización de la etiqueta *ELEMENT* indica que todo elemento de tipo *mensajes* está formado por cero o más elementos *mensaje*. Los calificadores que se pueden utilizar para indicar la repetición de los elementos son los siguientes:

TABLA 2. Sintaxis de etiqueta

	Tipo	Nombre	Modelo de datos	
<!	*ELEMENT*	*foro*	*(nombre,mensajes)*	>
<!	*ELEMENT*	*mensajes*	*(mensaje)+*	>
<!	*ELEMENT*	*titulo*	*(#PCDATA)*	>

TABLA 3. Sintaxis de *ATTLIST*

	Tipo	Nombre del elemento	Nombre del atributo	Modelo de datos	
<!	*ATTLIST*	*fecha*	*dia*	*CDATA #REQUIRED*	>

TABLA 4. VALORES DE *CDATA*

(valor1|valor2|···)	Lista de valores posibles
CDATA	Cadena de texto CDATA. Unparsed Character Data)
ID	Un identificador único que no es compartido por ningún otro atributo del tipo ID.
IDREF	Una referencia a un identificador definido en algún otro lugar del documento
IDREFS	Lista de referencias a identificadores separados por comas.
ENTITY	Nombre de una entidad definida en el DTD.
ENTITIES	Lista de entidades separadas por comas.
NMTOKEN	Un nombre XML válido (letras, números, guión, guión bajo y dos puntos).
NMTOKENS	Lista de nombres XML válidos separados por coma

TABLA 5. COMPORTAMIENTO DE ATRIBUTO

#REQUIRED	El atributo debe aparecer obligatoriamente.
#IMPLIED	No es obligatorio que el atributo aparezca.
"valor"	Valor por defecto que toma el atributo.
#FIXED "valor"	Valor que debe tomar obligatoriamente ese atributo.

- Signo de cierre de interrogación (?): cero o uno.
- Asterisco (*): cero o más.
- Signo de suma (+): uno o más.

Finalmente, en el último ejemplo estamos diciendo que el título es un cadena de texto codificada según el esquema de codificación que se indique en el documento *XML (PCDATA, Parser Character Data)*. La almohadilla (#) se utiliza para indicar que la palabra que viene a continuación es especial.

Además de los casos anteriores también se puede utilizar la barra horizontal (|) para indicar que un elemento puede tomar uno u otro valor.

La definición de un elemento también puede contener una de las dos siguientes palabras reservadas: *ANY* o *EMPTY*. La primera significa que el elemento puede estar constituido por otros elementos o simplemente almacena una cadena de texto (*PCDATA*). La palabra reservada *EMPTY* se utiliza para aquellas etiquetas que carecen de etiqueta de cierre, y por lo tanto, el elemento correspondiente no guarda nada, ni texto ni otros elementos.

ATRIBUTOS

La etiqueta *ATTLIST* se utiliza para expresar cómo han de ser los atributos de los distintos elementos, si es que los tienen. Podemos ver su sintaxis en la Tabla 3.

La primera palabra del modelo de datos de la etiqueta *ATTLIST* define el tipo de dato y puede tomar los valores indicados en la Tabla 4.

La segunda palabra en la parte correspondiente al modelo de datos sirve para indicar el comportamiento del atributo, según vemos en la Tabla 5.

Los documento *DTD* también permiten definir entidades.

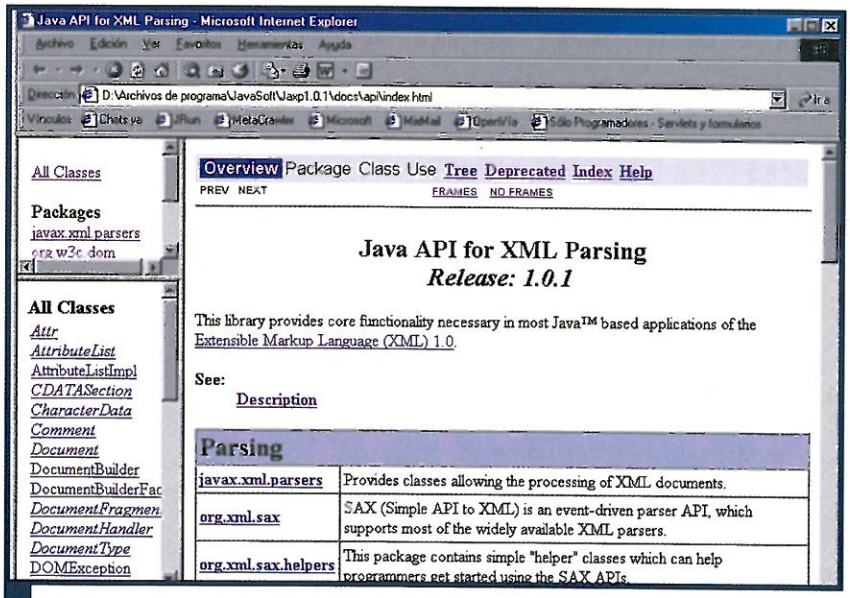


Figura 6- Página principal del documento de referencia del *API JAXP 1.0.1.*

TABLA 6. APLICACIÓN *PARSER* DE UNA FUENTE *XML*

```
import java.io.*;
import org.w3c.dom.*;
import javax.xml.parsers.*;
import org.xml.sax.*;

public class Ejemplo01 {
  public static void main (String argv []) {
    Document doc;
    try {
      DocumentBuilderFactory docBuilderFactory =
          DocumentBuilderFactory.newInstance();
      docBuilderFactory.setValidating(true);
      DocumentBuilder docBuilder =
          docBuilderFactory.newDocumentBuilder();
      docBuilder.setErrorHandler(
        new org.xml.sax.ErrorHandler() {
          public void fatalError(SAXParseException exception)
              throws SAXException { }
          public void error (SAXParseException e)
              throws SAXParseException { throw e; }
          public void warning (SAXParseException e)
              throws SAXParseException {
            System.out.println ("[SAXParseException Warning] " +
              "Línea " + e.getLineNumber() + " : " + e.getMessage ()); }
          }
      );
      doc = docBuilder.parse(new File("FORO.XML"));
      System.out.println("El nodo principal es..." +
          doc.getDocumentElement().getNodeName());
    } catch (SAXParseException e) {
        System.out.println("[SAXParseException] " + "Línea " +
          e.getLineNumber() + " : " + e.getMessage ());
    } catch (SAXException e) {
      System.out.println("[SAXException] " + e.getMessage());
    } catch (IOException e) {
      System.out.println("[IOException] " + e.getMessage());
    } catch (ParserConfigurationException e) {
      System.out.println("[ParserConfigurationException] " +
        e.getMessage());
    }
  }
}
```

ENTIDADES

Cuando en una página *HTML* queremos mostrar el carácter mayor que *(>)*, lo que hacemos es escribir la expresión *>*. Esto es lo que denomina una entidad. Los documentos *DTD* también pueden contener definiciones de entidades que serán utilizadas dentro del documento *XML*, o incluso dentro del propio *DTD*. Por ejemplo:

```
<!ENTITY copyr  "&#169;">
```

Lo anterior significa que en el documento *XML* correspondiente siempre podré utilizar la expresión *&copyr;* para significar el símbolo del *Copyright*.

EL API JAXP

Desde el sitio *Web* de *Sun* (*java.sun.com/xml/*) podemos descargar la *API JAXP*, así como la documentación correspondiente. Su instalación es completamente automática, y una vez que ha concluido, podemos empezar a utilizarla en nuestras aplicaciones *Java* para tratar fuentes de datos *XML*. Es importante recordar que antes de compilar y ejecutar una aplicación que utilice la *API JAXP*, hay que incluir en el *CLASSPATH* los ficheros *parser.jar* y *jaxp.jar* que se encuentran en el directorio principal que crea la instalación.

Se debe expresar la obligatoriedad de los atributos

La *API* cuenta con muchas clases, e iremos viendo casi todas poco a poco. Por el momento vamos a realizar un primer contacto haciendo simplemente una pequeña aplicación que revise una fuente de datos *XML* (Tabla 6).

En primer lugar, se crea una concreción de la clase DocumentBuilderFactory. Esta clase define una API que permite a las aplicaciones obtener el parser encargado de producir el árbol en forma objeto DOM que representa al documento XML. El método estático newInstance devuelve una instancia de la clase DocumentBuilderFactory.

La API JAXP permite tratar fuentes de datos XML

El método *setValidating* de la clase *DocumentBuilderFactory* recibe como único parámetro un valor

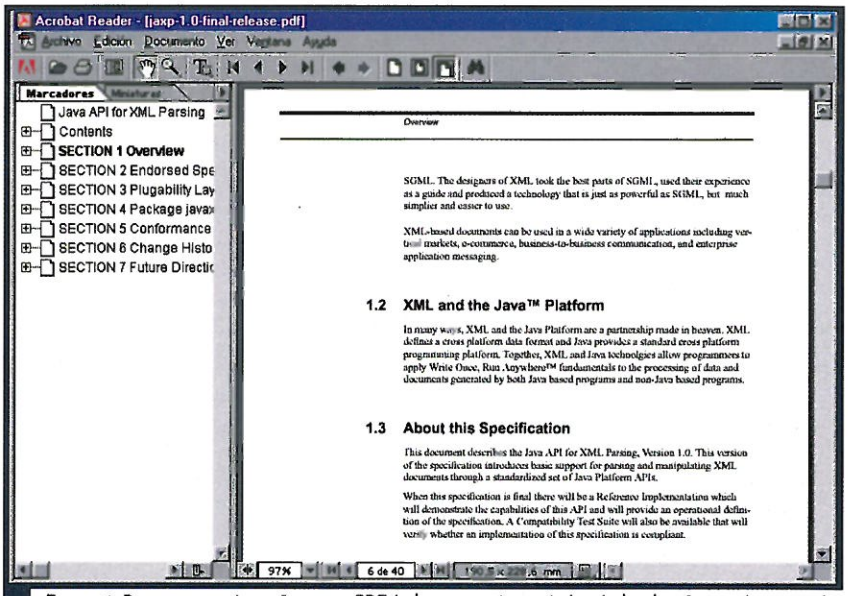

Acrobat Reader - [jaxp-1.0-final-release.pdf]
Archivo Edición Documento Ver Ventana Ayuda
Marcadores
Java API for XML Parsing
Contents
SECTION 1 Overview
SECTION 2 Endorsed Spe
SECTION 3 Plugability Lay
SECTION 4 Package javax
SECTION 5 Conformance
SECTION 6 Change Histo
SECTION 7 Future Directic
Overview
SGML. The designers of XML took the best parts of SGML, used their experience as a guide and produced a technology that is just as powerful as SGML, but much simpler and easier to use.
XML-based documents can be used in a wide variety of applications including vertical markets, e-commerce, business-to-business communication, and enterprise application messaging.
1.2 XML and the Java™ Platform
In many ways, XML and the Java Platform are a partnership made in heaven. XML defines a cross platform data format and Java provides a standard cross platform programming platform. Together, XML and Java technologies allow programmers to apply Write Once, Run Anywhere™ fundamentals to the processing of data and documents generated by both Java based programs and non-Java based programs.
1.3 About this Specification
This document describes the Java API for XML Parsing, Version 1.0. This version of the specification introduces basic support for parsing and manipulating XML documents through a standardized set of Java Platform APIs.
When this specification is final there will be a Reference Implementation which will demonstrate the capabilities of this API and will provide an operational definition of the specification. A Compatibility Test Suite will also be available that will verify whether an implementation of this specification is compliant.
97%
6 de 40

Figura 7- Documentación en formato *PDF* de la extención estándar de la plataforma Java que da soporte a la tecnología *XML*.

booleano. Si es *true* significa que se desea que el documento sea validado por el parser. Recordemos que el parser necesariamente analizará si el documento *XML* está bien formado, pero la validez es algo que va unido a la presencia de un *DTD*.

El método *newDocumentBuilder* crea una instancia de la clase *DocumentBuilder*. Si no puede completar la operación con éxito lanza una excepción del tipo *ParserConfigurationException*.

La clase *DocumentBuilder* define una *API* mediante la que se puede obtener el árbol en forma de objeto *DOM* que representa al documento *XML*, concretamente un objeto del tipo *org.w3c.dom.Document*. El *parsing* puede hacerse a partir de diversos orígenes, como muestra las distintas definiciones del método *parse*.

- *public Document parse(java.io.File f) throws SAXException, java.io.IOException*
- *public abstract Document parse (InputSource is) throws SAXException, java.io.IOException*
- *public Document parse (java.io.InputStream is) throws SAXException, java.io.IOException*
- *public Document parse(java.lang.String uri) throws SAXException, java.io.IOException*

El método *setErrorHandler* permite definir el manejador de eventos que será utilizado durante la fase del *parsing*. En el caso concreto que nos ocupa, se crea una nueva clase dentro de la propia llamada al método. Los métodos *fatalError*, *error* y *warning* serán invocados por el propio *parser* si se produce algún tipo de inconsistencia en la fuente de datos *XML*.

El método parser analiza el origen de datos XML

La interfaz *Document* representa el documento *XML*. Conceptualmente, es la raíz del árbol de datos y es "la puerta de entrada" a través de la cual accedemos a los datos.

El método *getDocumentElement* proporciona acceso al elemento o nodo que ocupa la raíz del árbol de datos. Por ejemplo, si se tratara de una página *HTML* que está siendo tratada como un documento *XML*, el nodo que devolvería este método sería el correspondiente a la etiqueta *<HTML>*.

El ejemplo anterior resume una primera aproximación a la utilización de la *API JAXP*. En sucesivos artículos continuaremos profundizando en su uso.

DIRECCIONES DE INTERÉS

Java(TM) Technology & XML
java.sun.com/xml/

Free XML tools and software
www.garshol.priv.no/download/xmltools/

Xml.com
www.xml.com

XMI Toolkit another alphaWorks technology
www.alphaworks.ibm.com/tech/xmitoolkit

Visual XML Tools
www.alphaworks.ibm.com/tech/visualxmltools

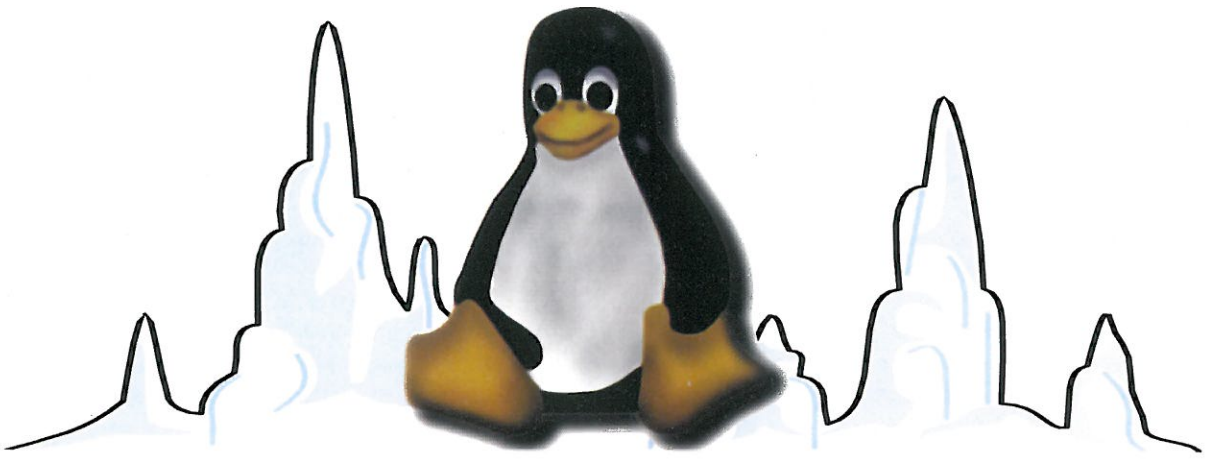
CUESoft EXml
www.garshol.priv.no/download/xmltools/prod/EXml.html

ezDTD
www.geocities.com/SiliconValley/Haven/2638/ezDTD.htm

PARA NO QUEDARSE HELADO CUANDO HABLEN DE LINUX

Incluye 2 CD's

DISTRIBUCIONES: RedHat 6.2: en castellano
X-WINDOW: Enlightenment: el espectáculo debe continuar
SISTEMA OPERATIVO: Mandrake 7.0: merece la pena
SISTEMA OPERATIVO: Partiendo el disco
DISTRIBUCIONES: Hispafuentes 7.0:
el inglés dejó de ser la excusa
PROGRAMACIÓN GRÁFICA: Programación multimedia con SDL
UTILIDADES: Edición de documentos en LaTeX: LyX
HERRAMIENTAS: Diseño fotográfico con Gimp



Curso de Programación con WAP (IV)

Constantino Sánchez Ballesteros.
Técnico Superior de Desarrollo Aplicaciones Informáticas.

El uso de *scripts* para *WML* en la creación de contenido *WAP*, nos ofrece una gran flexibilidad. Veamos cuales son sus posibilidades.

ALTERNATIVAS

El lenguaje *WMLScript* forma parte de una de las capas de *WAP*, y podemos usarlo para crear procedimientos del lado del cliente (recordemos que hay cliente "teléfono móvil" y servidor). Este lenguaje se basa en *ECMAScript*, pero se ha modificado para tener mejor soporte en dispositivos con bajo ancho de banda, como es el caso de los teléfonos móviles.

WMLScript ofrece nuevas funcionalidades que no soporta nativamente WML

Podemos utilizar conjuntamente *WMLScript* y *WML* o *scripts* de forma independiente. Para la transmisión de datos de los *scripts* se utiliza el formato binario. Esto permite aprovechar los canales de comunicación completamente y preservar la memoria requerida por el cliente.

Debido a las amplias características del lenguaje *ECMAScript*, se han "recortado" partes del lenguaje para hacer el *WMLScript* más pequeño y fácil de compilar a *bytecode*. Además, esto implica más facilidad para programar.

Para acabar con las características principales de *WMLScript*, diremos que se trata de un lenguaje de procedimiento, que soporta librerías estándar instaladas de forma local. En el siguiente ejemplo, vemos la definición de una función en *WMLScript*:

```
extern function getInfoDate
(variable) {
  WMLBrowser.setVar(variable,
    "Enero 1 2000");
  WMLBrowser.refresh();
}
```

Esta función, simplemente pasa una fecha al programa principal, mediante el retorno sobre la variable que pasamos a la función (*variable*).

Como ya se ha dicho, *WMLScript* se diseñó para tener soporte de *scripting* en la arquitectura *WAP*. Con *WML* podemos crear contenido como texto, imágenes, listas de selección, etcétera. Hay que tener en cuenta que todo el contenido será estático, y no es posible ampliar las características del lenguaje sin modificar el *WML*.

Debido a esas limitaciones, se creó *WMLScript*, de forma que muchos de los servicios implementados por los móviles pueden efectuarse con *WML*. Sumando a esto

TABLA 1. Tareas no soportadas nativamente por WML.

- Chequeo de validación de datos por el usuario.
- Facilidades de acceso para el usuario: llamadas telefónicas, envío de mensajes y agregar números de teléfono a la agenda o a la tarjeta *SIM*.
- Generar mensajes y diálogos de forma local, mostrando mensajes de alerta, de error, confirmaciones, etcétera.

la potencia de los *scripts* para determinadas tareas, podremos crear contenidos mucho más completos y de más calidad. Podemos usar el *scripting* para soportar funciones avanzadas en la interfaz de usuario, implementar "inteligencia" en los móviles, acceder a sus periféricos, y reducir el ancho de banda requerido para enviar datos entre el servidor y el cliente.

EL INTÉRPRETE DE WMLSCRIPT

El formato texto del *WMLScript* se compila a un formato binario antes de que pueda ser interpretado por *WMLScript*. El compilador de *WMLScript* codifica una unidad de compilación *WMLScript* a *bytecode*.

Cada una de estas unidades contiene *pragmas* (como en *C*) y un número determinado de funciones *WMLScript*. El compilador toma una unidad de compilación como entrada, y genera el *bytecode* del *script* como salida.

ARQUITECTURA DEL INTÉRPRETE

El intérprete de *WMLScript* toma el *bytecode* como entrada y ejecuta las funciones codificadas cuando éstas son llamadas. En la Figura 2 encontramos las principales partes encargadas de interpretar el *bytecode* generado en un *script WMLScript*.

Cada función codificada establece el número de parámetros que acepta y las instrucciones utilizadas en la tarea diseñada por el programador. Para llamar a una función de *WMLScript* hay que especificar su nombre, sus argumentos, y la unidad de compilación en la que se encuentra declarada. Una vez completada la llamada, el intérprete de *WMLScript* devuelve el control y los valores pertinentes al punto del programa en que se hizo la llamada.

Cuando una función se está interpretando, el intérprete de *WMLScript* mantiene el siguiente estado de información:

1. *IP (Puntero a la Instrucción)*: apunta a la instrucción del *bytecode* que está siendo interpretada.

2. *Variable*s: parámetros de la función y variables.

3. *Operand stack*: se utiliza para evaluar la expresión y pasar argumentos entre las funciones llamadas y el programa principal que las llama.

4. *Function call stack*: una función de *WMLScript* puede llamar a otras funciones de la misma unidad o de otra distinta, además de llamar a funciones de librerías. *function call stack* guarda la información de las funciones y sus direcciones de retorno.

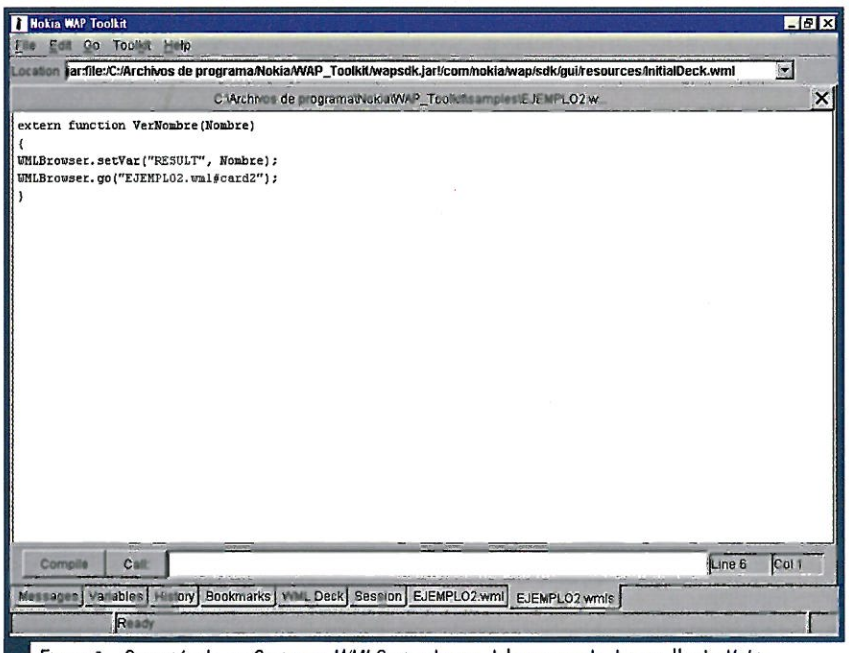


Figura 1. - Creación de un *Script* en *WMLScript* dentro del entorno de desarrollo de *Nokia*.

WMLScript es un lenguaje fundamentado en scripting

ACCESO A LAS FUNCIONES DE WMLSCRIPT

WMLScript tiene dos mecanismos para controlar el acceso a las

funciones de la unidad de compilación, consistentes en una clave externa, y un *pragma* específico. Además, el intérprete de *WMLScript* debe soportar las siguientes características:

- *Funciones externas*: Sólo éstas pueden ser llamadas desde otras unidades compiladas.
- *Control de acceso*: El acceso a funciones externas definidas en una unidad de compilación desde otras unidades sólo es posible teniendo las mismas definiciones de rutas de acceso y dominio.

SET DE CARACTERES

El intérprete de *WMLScript* sólo debe utilizar un conjunto de caracteres *(set de caracteres nativo)* para todas las operaciones de cadenas (*strings*). Para utilizar el *set* de caracteres nativo tenemos que utilizar la función *Lang* de la librería estándar:

```
Lang.characterSet()
```

URL'S

La *Web* es una red de bases de datos y dispositivos donde se establecen tres áreas de especificaciones para asegurar la interoperabilidad:

- Modelo de notación unificado: esta notación se implementa mediante las *URL*, que permiten una forma estándar de nombrar cualquier recurso colgado en la red.
- Protocolos estándar para transportar la información: por ejemplo, *HTTP*.
- Tipos de contenido estándar: como *HTML* o *WMLScript*.

La unidad de compilación *WMLScript* se nombra utilizando *URL's* y puede obtenerse sobre protocolos estándar con la semántica *HTTP*.

En *WMLScript*, las *URLs* se utilizan cuando el agente de usuario (móvil) tiene que hacer una llamada a un *script* de *WMLScript*, cuando se especifican diferentes unidades de compilación externas, y cuando se especifica información al control de acceso.

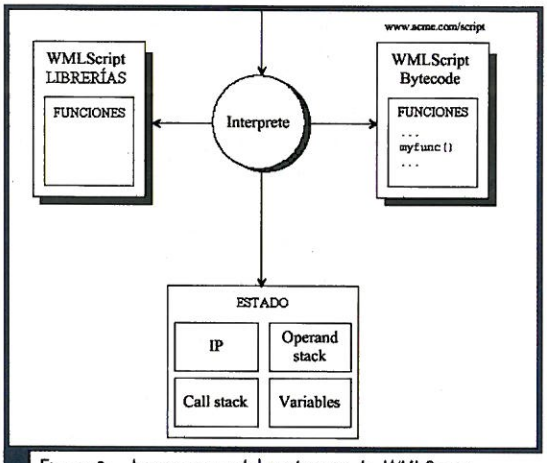


Figura 2. - Arquitectura del intérprete de *WMLScript*.

FRAGMENTOS

WMLScript adopta el método de *HTML* para nombrar localizaciones desde un recurso. Un "*fragment anchor*" de *WMLScript* se especifica por la *URL* del documento seguida del símbolo "almohadilla" (#) y por un identificador de fragmento.

WMLScript utiliza los *fragment anchors* para identificar funciones *WMLScript* individuales dentro de una unidad de compilación.

En el siguiente apartado veremos la sintaxis del *fragment anchor* con un ejemplo.

PASO DE PARÁMETROS

El agente de usuario puede llamar a una función externa de *WMLScript* estableciendo la siguiente información con *URLs* y *fragment anchors*:

Para la *URL* de la unidad de compilación.

```
http://www.solopro.es/Scripts.scr
```

Para el nombre de la función y los parámetros.

```
testFunc('Test%20argument', -8)
```

El WMLScript se basa en funciones

La anotación final de la *URL* con el fragmento quedaría así:

```
http://www.solopro.es/Scripts.scr#
    testFunc('Test%20argument', -
    8)
```

Si la *URL* anterior retorna una unidad válida de compilación:

1. Se chequea el control de acceso. La llamada a la unidad de compilación fallará si ésta no se ha hecho correctamente
2. El nombre de la función del *fragment anchor* se comprueba con

TABLA 2. Sintaxis para los diferentes espacios en blanco y terminadores de línea

ESPACIOS EN BLANCO
<TAB>
<VT>
<FF>
<SP>
<LF>
<CR>
TERMINADORES DE LÍNEA
<LF>
<CR>
<CR><LF>

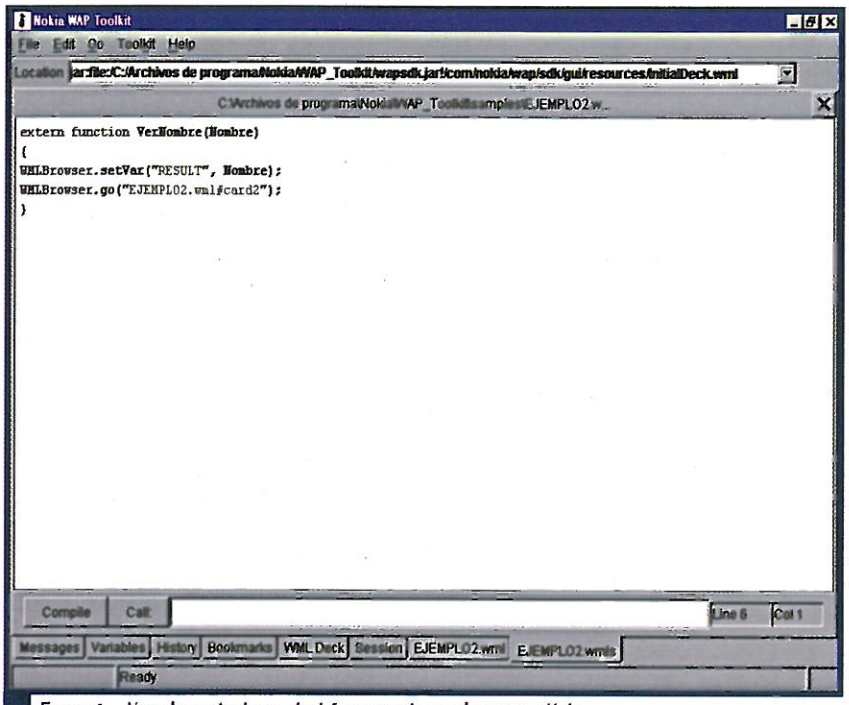


Figura 3. - Visualización de un *deck* formateado en el entorno *Nokia*.

las funciones externas de la unidad de compilación. La llamada fallará si no está definida la función.

3. Del mismo modo, se comprueban los argumentos pasados a la función. También encontraremos un error si variamos el número de parámetros o su sintaxis no es válida.

Al crear unidades de compilación, se establecen dos tipos de archivos: Forma textual (tal y como editamos el código): *text/vnd.wap .wmlscript*, y forma binaria (unidad ya compilada): *application/vnd.wap .wmlscriptc*

TABLA 3. Secuencias de escape especiales.

\' Apóstrofe
\" Comilla doble
\\ Backslash
\/ Slash
\b Backspace (retroceso de caracter)
\n Nueva línea
\r Retorno de carro
\t tabulación horizontal

SOBRE EL LENGUAJE

En *WMLScrip,t* al igual que en *C*, no es lo mismo la variable *a* que la variable *A*. Un buen consejo: mayúsculas para variables y nombres de las funciones. Por supuesto, eso dependerá de la forma de programar de cada uno.

WMLScript omite los espacios, tabulaciones, nuevas líneas, etcétera que aparecen entre *tokens*, excepto aquellos que forman parte de constantes de tipo cadena.

Para crear comentarios, *WMLScript* incluye dos instrucciones:

1. Los comentarios de línea comienzan con // y terminan con el final de la línea.

2. Los comentarios para bloques de código consisten en múltiples líneas que comienzan con /* y acaban con */.

Figura 4. - Un *script* en ejecución.

Los lectores familiarizados con *C/C++* verán que es la misma sintaxis.

WMLScript soporta cuatro tipos de literales:

1. *Integer* (entero)
2. *floating-point* (coma flotante)
3. *string* (cadena de caracteres)
4. *Boolean* (verdadero/falso)

Los literales de tipo *Integer* pueden representarse en tres formas: Decimal (*0 1 2 3 4 5 6 7 8 9*), Octal:(*0*), y exadecimal (*0x*).

Los literales en coma flotante pueden contener un exponente y un punto decimal. El exponente debe indicarse con las letras *e* o *E*.

Variables, funciones, y pragmas son los tres tipos de funciónes de WMLS

Los literales de tipo cadena de caracteres (*string*) pueden contener cualquier secuencia de cero o más caracteres encerrados entre comillas dobles "" o simples ''. Dado que *WMLScript* no puede representar algunos caracteres como *strings*, existen secuencias de escape especiales para ello.

Ejemplos de *strings* válidos:

```
"cadena"
"Comillas: \""
```

IDENTIFICADORES

Podemos usar identificadores para nombrar y referirnos a los tres tipos de elementos de *WMLScript*: variables, funciones, y *pragmas*. Los nombres de identificadores no pueden comenzar con un dígito, pero sí con guión bajo _. Aquí vemos algunos ejemplos de identificadores legales:

```
horaDelDia
velocidad
calidad
MI_DIRECCION
var0
_miNombre
```

Las letras mayúsculas y minúsculas, diferencian unos nombres de otros, por eso, la variable *velocidad* no es igual que *Velocidad*.

Veamos algunos identificadores ilegales que devuelven error al compilar:

```
while
for
if
mi~nombre
$sys
123
3piezas
probar.esto
```

Podemos utilizar los mismos nombres de identificadores para diferentes usos, como nombres de función, variables o *pragmas*. En el siguiente ejemplo se muestra un detalle de esta característica:

```
use url myTest
     "http://www.SoloP.com/script"
     ;
function myTest(myTest) {
var value =
     myTest#myTest(myTest);
return value;
};
```

VARIABLES Y TIPOS DE DATOS

Una variable es un nombre asociado con un valor. Podemos utilizar variables para guardar y manipular datos del programa. *WMLScript* soporta variables locales sólo cuando son declaradas dentro de funcio-

TABLA 4. Asignaciones con operaciones.

OPERADOR	OPERACIÓN
=	Asignación
+=	Incrementa números, concatena cadenas y asigna
-=	Disminuye asigna
*=	Multiplica y asigna
/=	Divide y asigna
div=	Divide (división entera) y asigna
&=	Operación *and* y asigna
^=	Operación *xor* y asigna
|=	Operación *or* y asigna

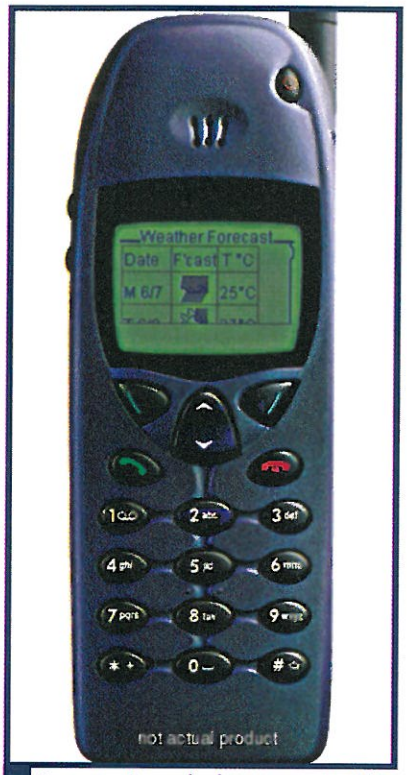


Figura 5. - Pese a las limitaciones de *WAP*, podemos crear páginas de cierta complejidad.

nes, o pasadas como parámetros de las funciones.

Para declarar una variable utilizaremos la clave *var* y el nombre de la variable.

```
var x;
var precio;
var x,y;
var cantidad = 3;
```

Hay que tener en cuenta que las variables deben ser declaradas antes de poder utilizarlas. Su activación es opcional, y en caso de no haberlas activado, contendrán de forma automática una cadena vacía ("").

El rango de acción de una variable, así como su permanencia en memoria, depende de la función en la que ha sido declarada. Todos los nombres de variables contenidos en una función deben ser únicos:

```
function testPrecio (Precio) {
  if (Precio > 100) {
    var Nuevo_Precio = Precio;
  }
  else {
   Nuevo_Precio = 100;
};
return Nuevo_Precio;
};
```

El tiempo de vida de una función varía desde que es declarada hasta que termina la función:

```
function Testing() {
  x = 1; // Error: uso de varia-
    ble antes de ser declarada
  var x,y;
   if (x) {
     var (y); // Error: redecla-
    ración
   };
};
```

Sólo es posible acceder a variables desde la función donde han sido declaradas. El acceso al contenido de una variable se realiza con su nombre:

```
var miEdad = 37;
var tuEdad = 63;
var nuestraEdad = miEdad +
    tuEdad;
```

OPERADORES Y EXPRESIONES

Aquí vamos a ver algunos de los operadores soportados por *WMLScript*, y cómo usarlos para crear expresiones complejas.

Podemos utilizar los mismos nombres de identificadores para diferentes usos

WMLScript permite varias formas de asignar un valor a una variable. La más simple consiste en utilizar el signo =, pero también soporta asignaciones con operaciones (Tabla 4).

Soporta además, todas las operaciones aritméticas básicas.

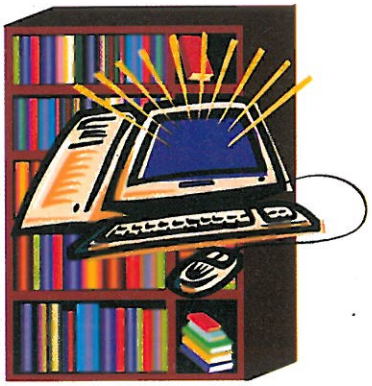
Veamos algunos ejemplos válidos de instrucciones utilizados por los operadores descritos en la Tabla 5:

```
var y = 1/3;
var x = y*3+(++b);
```

En el próximo número, seguiremos estudiando el uso de funciones, la creación de bucles, utilización de operadores, acceso a funciones de librerías, etcétera.

TABLA 5. Operadores aritméticos más utilizados.

OPERADOR	OPERACIÓN
+	Suma números y concatena cadenas de caracteres.
-	Resta.
*	Multiplica
/	Divide.
<<	Rota valor binario hacia la izquierda.
>>	Rota valor binario hacia la derecha.
--	Pre/pos decremento
—	Pre/pos incremento
~	Operador *NOT*



FAST.lib 2000 para Windows

Manu Roibal.
Ingeniero Superior de Informática.

Hace ya mucho que los gurús de la Informática vaticinaron que el arte de programar consistiría en amasar un conjunto de componentes de diferentes fabricantes y hacer que todo ello, junto pero no revuelto, funcionase. Y lo cierto es que, en la actualidad, la situación se parece mucho a aquella previsión.

Una aplicación actual, contiene muchas menos líneas de código que las más comunes de las de hace una década (al menos para hacer el mismo trabajo), ya que gran parte de sus funciones son asumidas por esos entes pseudo-divinos denominados controles.

FAST.lib 2000 para *Windows* es una biblioteca de controles y clases ActiveX desarrollada por *FAST Software* y distribuida en Internet en régimen de *Freeware*. Este producto, enteramente español ya va por la versión 1.3.2., contando con más de 7.000 usuarios a lo largo y ancho del mundo.

La instalación está soportada por un asistente que guía al usuario durante todo el proceso, simple y claro. *FAST.lib 2000* para *Windows* soporta entornos de cualquier plataforma *Windows* de 32 *bits*: *Windows 9x, Windows NT 3.51* y *NT 4.0*, y *Windows 2000*.

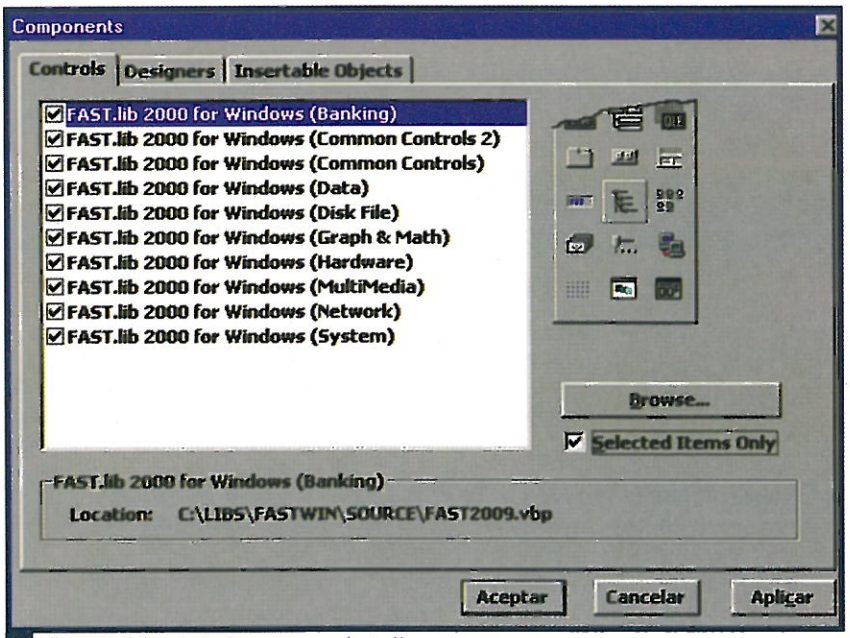


Figura 1.- Librería de Componentes *ActiveX*.

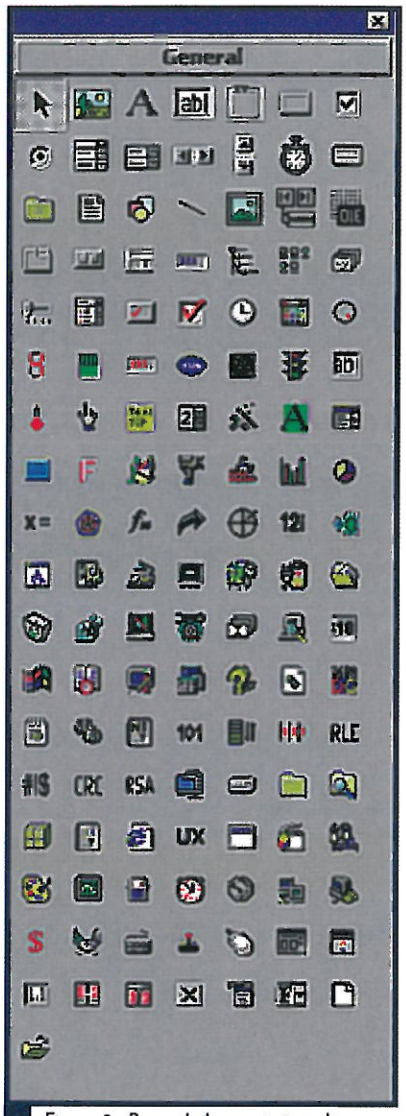


Figura 2.- Barra de herramientas de *FAST.lib 2000 for Windows.*

Básicamente, los requerimientos físicos son los impuestos por el *SO* y el entorno de programación donde se ejecuten. Por ello, y según *SO* soportados, éstos no deberían ser inferiores a un *Pentium* a 133 *Mhz* con 16 *Mb* de memoria *RAM*. Los requerimientos de espacio en disco son de 8 *MB*.

Actualmente, tiene más de 100 controles para hacer prácticamente de todo, desde mostrar un gráfico *3D* o analizar el contenido de un *ZIP*, hasta mostrar un botón con forma elíptica.

Pero vayamos por partes. La biblioteca está dividida en 11 componentes *ActiveX* según se muestra en la Figura 1.

COMPONENTES

No es que la librería sólo tenga 11 controles, pues como dijimos, éstos son más de 100. La Figura 2 muestra la barra de herramientas de *Visual Basic* tras añadir los controles de *FAST.lib 2000* para *Windows* a un proyecto.

Probablemente, los dos componentes más llamativos, serán *Common Controls* y *Common Controls 2*. Estas *OCX* incorporan 30 controles *ActiveX* (*checks*, barras de progreso, controles de fechas, botones, controles de colores y fuentes, termómetros...) Todos los controles mostrados en la Figura 3 son *WYSIWYG*, es decir, funcionan en diseño, igual a como el usuario los verá cuando la aplicación esté en producción.

Hay controles para casi todo, desde un reloj analógico, cajas de

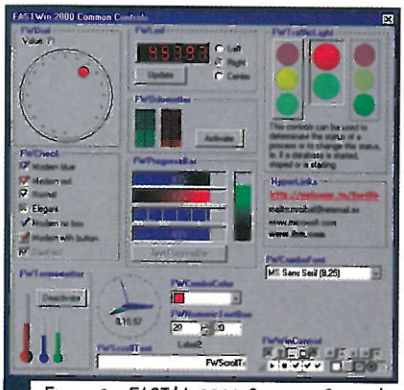


Figura 3.- *FAST.lib 2000 Common Controls.*

texto numéricas con *spin button*, *combos* de fuentes, hipervínculos, *checkboxes*, termómetros, controles de volumen, *leds*, semáforos...

FAST.lib 2000 para *Windows* dispone de una galería con tres tipos de botones: *FWButton* que muestra botones como los de *Borland*, *FWRealButton*, con botones de forma elíptica y foco de luz, y *FWCoolButton*, que muestra botones que se activan cuando el ratón pasa sobre ellos.

En cuanto a la implementación de los controles de tipo etiqueta o *Label*, *FAST.lib 2000* aporta dos tipos perfectamente diferenciados, esto es, *FWLabel* y *FWLabel-*

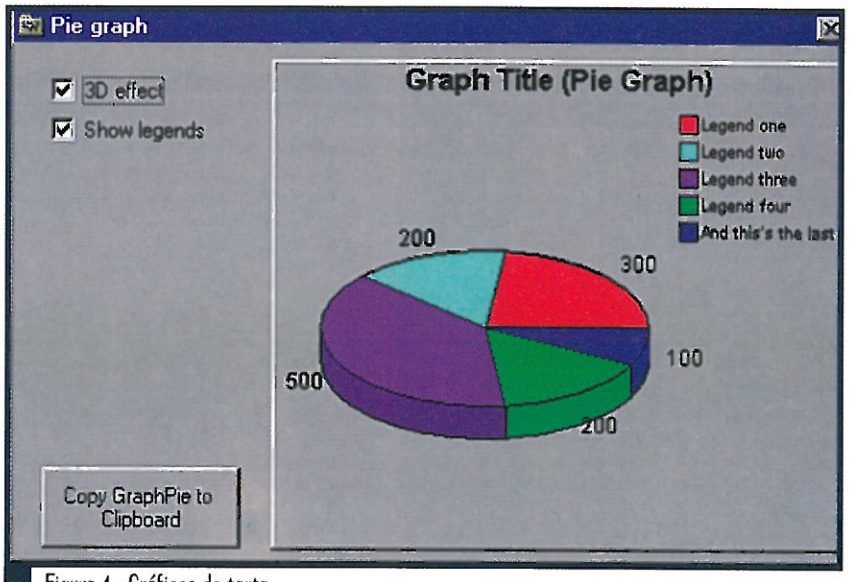


Figura 4.- Gráficos de tarta.

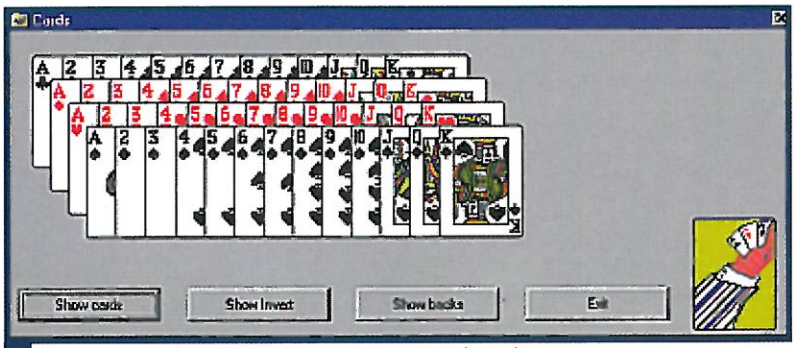


Figura 5.- *Fast.lib 2000* soporta naipes y dados en las aplicaciones.

3D. El primero de ellos permite hacer fundidos de color en la etiqueta al mismo tiempo que permite (en el caso de fuentes *TrueType*) hacer un trazado del texto mediante una imagen de referencia. El control *FWLabel3D* permite crear textos con apariencia *3D* con cualquier combinación de color

En cuanto a controles de entrada de datos, *FAST.lib 2000* implementa varios controles muy interesantes, como los de fechas, fechas en formato combo, una caja de texto de introducción de direcciones *IP*, un *combo box* y una caja de texto capaz de completar textos, al igual que lo hace la barra direcciones de *Internet Explorer*, un combo de colores... Probablemente este es el punto de la biblioteca en el que se está trabajando más y mejor, a fin de incorporar nuevas funcionalidades a los controles ya existentes, a la vez que controles en la galería.

La descripción de las propiedades, métodos y eventos de estos controles sería larguísima, por lo que escapa del objeto de este artículo. No obstante, se puede decir que, como es obvio, todos los controles disponen de los eventos habituales, como *Change, GotFocus, LostFocus, Click, DblClick, MouseDown/Move/Up, KeyPress, KeyDown/Up*... así como los eventos específicos de la funcionalidad de cada control. Lo mismo es aplicable para los métodos y propiedades.

GRÁFICOS

La parte gráfica de la librería esta perfectamente cubierta por varios controles, entre los que destacan *FW GraphPie, FWGraphBar, FWCurves2D* y *FWMan delbrot*. Éstos, y 8 más, conforman *FAST.lib 2000 for Windows (Graph & Math)*, es decir, la biblioteca que contiene las rutinas gráficas y matemáticas.

La Figura 4 muestra la apariencia del control *FWGraphPie*. Este control, al igual que el resto de los controles de *FAST.lib 2000*, destaca por su simplicidad. Como ejemplo, se muestra el código que fue necesario escribir para mostrar ese gráfico.

```
Private Sub Form_Load()
  With graPie
    Call .Create("Graph
Title (Pie Graph)")
Call .Add(300, vbRed, "Legend
one")
Call .Add(200, vbCyan,
"Legend two")
Call .Add(500, vbMagenta,
"Legend three")
Call .Add(200, vbGreen,
"Legend four")
Call .Add(100, vbBlue, "And
this's the last")
```

TABLA 1.

```
Private Sub Form_Load()
  Dim intInd  As Integer

  Call Randomize

  With objGraphBar
   Call .Create("Graph Title (Monthly)", 12)

   .ShowLegends = True

   .Style = FLWGraphMath.flwGraphBars3D

   ' serie 1
   With .Add(vbRed, "Serie one")
    For intInd = 1 To 12
     .Value(intInd) = Rnd * 500
    Next
   End With

   ' serie 2
   With .Add(vbGreen, "Serie two")
    For intInd = 1 To 12
     .Value(intInd) = Rnd * 400
    Next
   End With

   ' serie 3
   With .Add(vbBlue, "Serie three")
    For intInd = 1 To 12
     .Value(intInd) = Rnd * 300
    Next
   End With

   .Legends(1) = "January"
   .Legends(2) = "February"
   .Legends(3) = "March"
   .Legends(4) = "April"
   .Legends(5) = "May"
   .Legends(6) = "June"
   .Legends(7) = "July"
   .Legends(8) = "August"
   .Legends(9) = "September"
   .Legends(10) = "October"
   .Legends(11) = "November"
   .Legends(12) = "December"
  End With
End Sub
```

```
  End With
  graPie.ShowLegends = True
End Sub
```

La Figura 6 muestra un control de tipo *FWGraphBar* en ejecución. Al igual que en el caso anterior, se muestra el fragmento de código necesario para mostrar dicho gráfico, que como se verá, es realmente simple (Tabla 1).

El control *FWMandelbrot*, permite dibujar imágenes fractales, y hacer *zoom in* y *out* a cualquier nivel de profundidad. Por su parte, *FWCurves2D* permite dibujar cualquier tipo de curvas en el espacio *2D*, tanto en coordenadas polares como cartesianas. Conviene destacar que representa cualquier tipo de curva, ya que dispone de un motor de análisis de funciones en lugar de tener un conjunto de funciones conocidas de antemano. Esto, le permite dibujar cualquier curva, por compleja que sea. En el siguiente ejemplo se dibuja una conocida curva denominada *Cardioide*, que se representa mediante la función matemática *(n * (1 – cos(t)))*, en coordenadas polares.

En la Tabla 2 se ve cómo mostrar una curva en el control *FW Curves2D*.

TABLA 2.

```
Private Sub cmdRefresh_Click()
  Dim strCaption As String

  strCaption = Me.Caption
  Me.Caption = Me.Caption & ", Redrawing... please wait a moment"
  DoEvents
  With objCurves
   ' sets the values

   If optCoord(0).Value Then
     .CoordinateType = FLWGraphMath.flwCurves2DCoordPolars
   ElseIf optCoord(1).Value Then
     .CoordinateType = FLWGraphMath.flwCurves2DCoordCartesians
   End If

   .Axis = IIf(chkYAxis = 0, FLWGraphMath.flwCurves2DAxisX, _
           FLWGraphMath.flwCurves2DAxisY)
   .Points = Val(txtPoints)

   .Expression = txtExpression
   .VarName = txtVarName
   .VarFrom = Val(txtVarFrom)
   .VarTo = Val(txtVarTo)

   .DrawColor = vbBlue
   .LineWidth = 2

   ' draws the curve
   If .Error.Number <> 0 Then
     Call MsgBox("Error: " & .Error.Description)
   End If
  End With
  Me.Caption = strCaption
End Sub
```

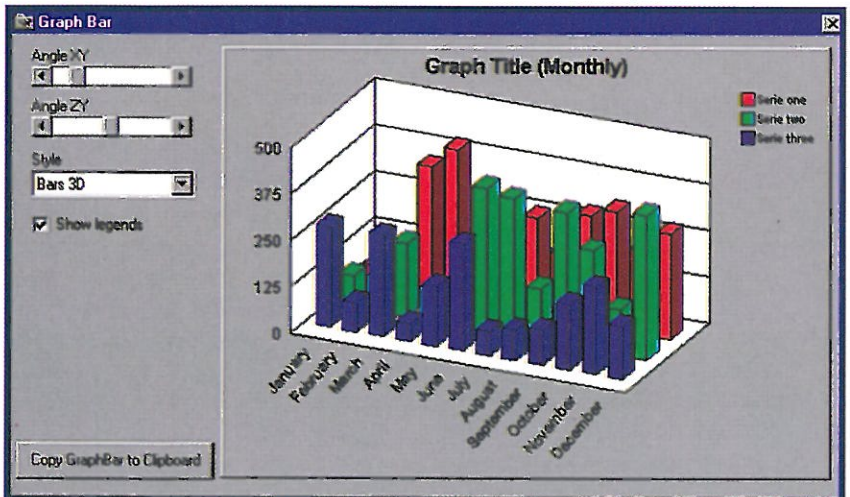


Figura 6.- Gráficos de barras, zonas, líneas, acumuladas...

LOS MENÚS

En cuanto a menús, *FAST.lib 2000* implementa un menú tradicional muy mejorado respecto al menú estándar de *Windows*, denominado *FWMenu*. Permite crear barras de menú en cualquier lugar de la ventana, de cualquier color y tipo de letra, y añadir imágenes a los *items* del menú. El control *FWActiveMenu* permite implementar menús similares a los que implementa *OutLook* o *Windows 2000*, es decir, menús verticales que hacen *scroll* para mostrar unas opciones u otras.

Todas las propiedades del control *FWMenu* pueden modificarse por código o en tiempo de diseño mediante el uso de la página de propiedades que muestra el control.

FAST.lib 2000 no se olvida de la parte lúdica de la vida. Por ello, implementa controles para permitir añadir soporte de naipes y dados a las aplicaciones: *FWCards* y *FW Roller*. Junto a ellos, en la sección multimedia, se incluyen controles para el manejo de archivos de ondas (*WAVE*) y archivos *MIDI*, manejo del *joystick* y todo aquello relacionado con las aplicaciones multimedia.

La Figura 5 muestra algunos de estos controles en ejecución. Obviamente, nada de lo mostrado se dibuja manualmente, ya que son los controles los que se encargan del refresco de las imágenes de forma automática.

TABLA 3.

```
Private Sub objNTLog_Change()
  Dim strType As String

  lblLabel(0) = "Event number " & objNTLog.EventNumber & "/" &
objNTLog.Count
  lblLabel(1) = "Generated on " & objNTLog.TimeGenerated
  lblLabel(2) = "Written on   " & objNTLog.TimeWritten
  lblLabel(3) = "Machine      " & objNTLog.ComputerName
  lblLabel(4) = "Category     " & objNTLog.EventCategory
  lblLabel(5) = "Event ID     " & objNTLog.EventID
  Select Case objNTLog.EventType
    Case flwEventError
      strType = "Error"
    Case flwEventWarning
      strType = "Warning"
    Case flwEventInfo
      strType = "Info"
    Case flwEventAuditSuccess
      strType = "Audit success"
    Case flwEventAuditFailure
      strType = "Audit failure"
  End Select
  lblLabel(6) = "Event type   " & strType
  lblLabel(7) = "Event source " & objNTLog.SourceName
  txtText = objNTLog.Text
End Sub
```

INFORMACIÓN DEL SISTEMA

Si lo que se desea es obtener información del sistema está de suerte, pues *FAST.lib 2000* obtiene toda aquella información que es técnicamente posible: analizar cualquier *DLL* o *EXE*, obtener información pura de la configuración sistema y el rendimiento, manejar los servicios de *NT* (detener, arrancar...), manejar el *log* de *NT*...

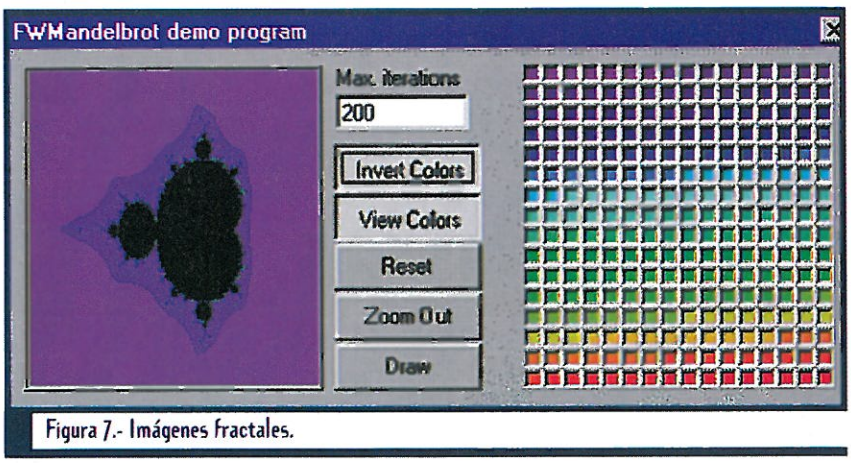


Figura 7.- Imágenes fractales.

FWServices permite obtener la lista de servicios del sistema (solo *NT y Windows 2000* por razones obvias) así como detener, arrancar, poner en pausa o configurar cualquier servicio. De esta forma, se dota a los programas *Windows* de todo el control sobre los servicios que necesitan para funcionar correctamente.

Como otra muestra de la simplicidad de uso de *FAST.lib 2000* para

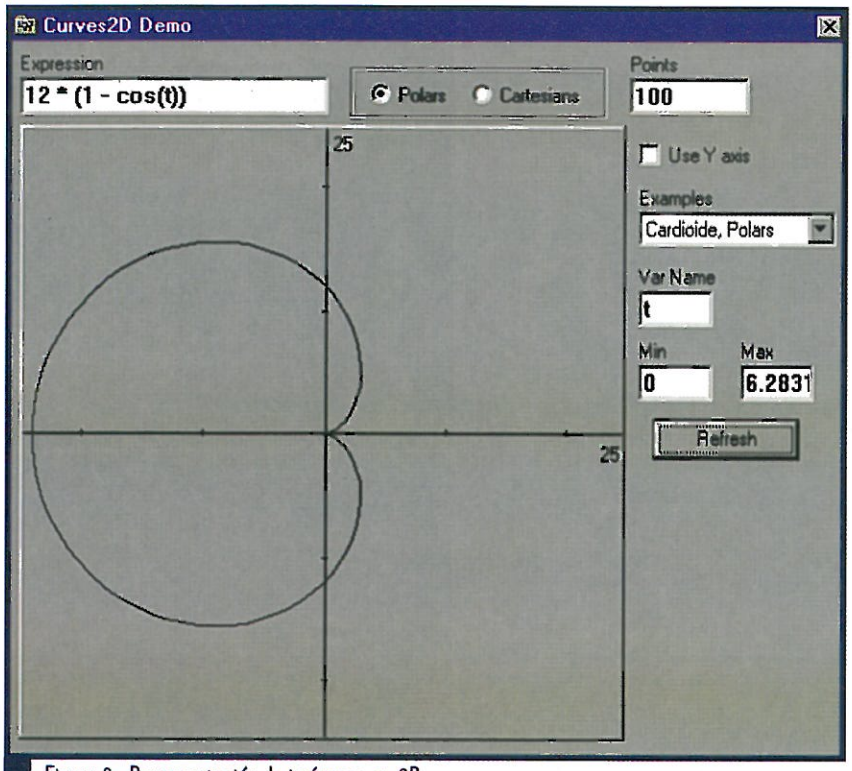


Figura 8.- Representación de imágenes en *2D*.

Windows, en la Tabla 3 se muestra un fragmento de código que permite acceder a *log* de Windows *NT*.

Por su parte, el control FWSystemInfo obtiene información del sistema en todos los niveles, desde el tipo de procesador al número de pinceles que soporta el subsistema de vídeo de *Windows*.

Además es posible obtener la versión del procesador y *SO*, información del registro, fecha y hora de la *BIOS*, información sobre la configuración de *Windows*, como el país, idioma, la configuración numérica y de fechas...

Pero la lista no termina aquí. Durante este artículo, se ha hablado sólo de de unos 20 controles aproximadamente, de entre todos los que *FAST.lib 2000* para *Windows* pone a nuestra disposición, además de las 120 clases *ActiveX*.

Hay controles para manejar consolas *DOS* desde *Windows*, obtener información de los archivos y discos, manejo de unidades y asignaciones de red, manejo de cuentas de usuario en *Windows NT* y *2000*, compresión de datos *LZ* y *RLE*, codificación con claves *RSA*, conversiones de datos, manejo de *bits*, gestión y configuración de fuentes *ODBC*, gestión de los archivos y subsistema de ayudas, cálculo de préstamos bancarios, gestión y validación de tarjetas de crédito, manejo de *stacks*, cálculo de *CR C32*, gestión de archivos *ZIP* y *ARJ*, gestión y acceso al registro, soporte de sistemas multimonitor, gestión de la batería (para portátiles), gestión de la papelera de reciclaje y mucho, muchísimo más.

Sin duda, con *Fastlib 2000*, la programación encuentra en esta galería de *ActiveX* -la mayor del mercado- nuevas posibilidades que harán la labor de los progamadores más cómoda y productiva.

Y lo mejor de todo. es que es gratis. *FAST.lib 2000* para *Windows* está disponible para descargas en la dirección http://welcome.to/fastlib. Y aunque la *Web* esté en inglés, no se equivoque, es un producto *made in Spain*.

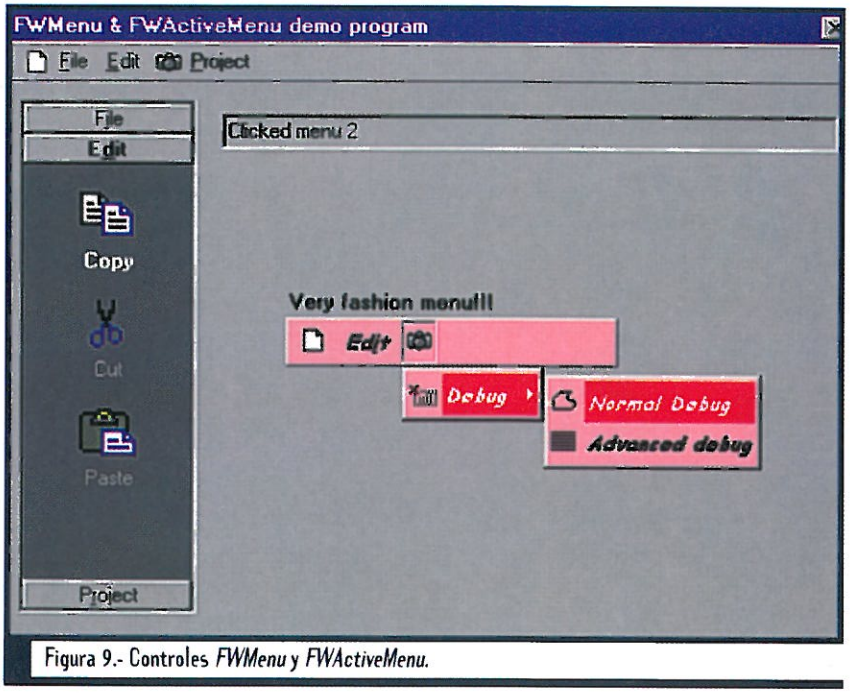


Figura 9.- Controles *FWMenu* y *FWActiveMenu*.

¿TE FALTA ALGÚN NÚMERO DE...?

LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SÓLO PROGRAMADORES

ATENCIÓN

Los números 49 al 51, ambos inclusive, se encuentran AGOTADOS

LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO
SÓLO PROGRAMADORES
PROGRAMACION GRAFICA
Sacar todo el rendimiento a una Voodoo 2 con las librerías 3D Glide
DESARROLLO CORBA
Marcando la diferencia entre la norma CORBA y DCOM
HTML DINAMICO
El efecto Scrolling
INTERNET
Cómo crear un buscador
JAVA
Construcción de aplicaciones gráficas multiplataforma con AWT
SEGURIDAD
Un cortafuegos con FireWall Toolkit
MULTIMEDIA
Reconocimiento de voz: DictaPad

LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO
SÓLO PROGRAMADORES
SONIDO Y VÍDEO CON JAVA (II)
APLICACIONES CON VÍDEO CONFERENCIA (III)
APLICACIONES IIS EN VISUAL BASIC (Y II)
Captura de vídeo con DELPHI
GLIDE (Y VII)
BASES DE DATOS CON ORACLE Y SQL SERVER
DIRECTX 6.1 (II)
BORLAND C++ BUILDER 4
XML (II)

BOLETÍN DE PEDIDOS

Rellene o fotocopie el cupón y envíelo a REVISTAS PROFESIONALES, S.L.
(Revista Sólo Programadores). C/ San Sotero, 5. 1ª Planta. 28037 Madrid
Tlf: 91 304 87 64. Fax: 91 327 13 03

Deseo que me manden los número/s.....
.....
.....

NOMBRE Y APELLIDOS:.....
.....

EDAD:..... PROFESIÓN: .....TFNO: .....
DOMICILIO: .....
CIUDAD: .....
PROVINCIA: .....C.P: .....

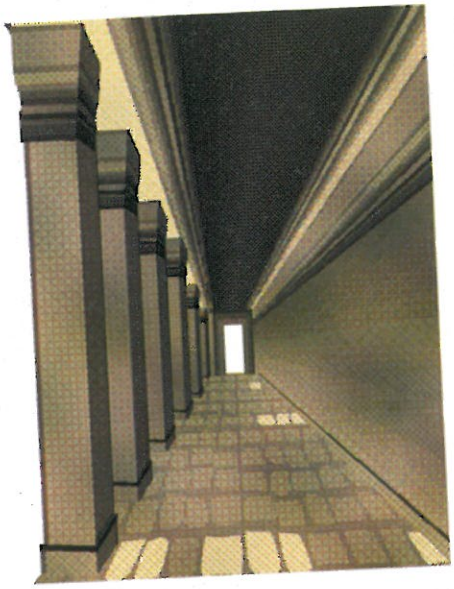
❑ **Precio por unidad: 975 pesetas** + 700 ptas. de gastos de envío.

FORMAS DE PAGO:

❑ Giro postal a nombre de REVISTAS PROFESIONALES, S.L
❑ Talón bancario a nombre de REVISTAS PROFESIONALES. S.L
❑ Domiciliación bancaria
❑ Contra reembolso

DATOS DE DOMICILIACIÓN:

Banco..... FIRMA
Domicilio.....
Nº de Cuenta.....|.....|.....|.....
Titular.....
Fecha.....



Programación orientada a objetos (y V)

Jordi Agost Moré.
Profesor de Programación Multimedia de la Universidad de Lleida.

En este artículo, el último de la serie, examinaremos dos ejemplos ilustrativos de la implementación de dos clases y su uso en nuestros programas.

Presentamos dos ejemplos para mostrar las posibilidades de la *POO* dentro de *Visual Basic*. Primero veremos una clase que utilizaremos para tratar el acceso de datos de los objetos *ADO*, y en segundo lugar, una clase que servirá para acceder a los ficheros de inicio.

UNA CLASE PARA OBJETOS ADO

En este ejemplo vamos a encapsular unas cuantas tareas que encontraremos siempre que hagamos un programa que incluya una *BBDD*.

Lógicamente, para usar esta ... que activar las librerías de los objetos de datos *ActiveX* (*ADO* o *ActiveX Data Objects*) en la versión más actualizada de nuestras referencias del proyecto.

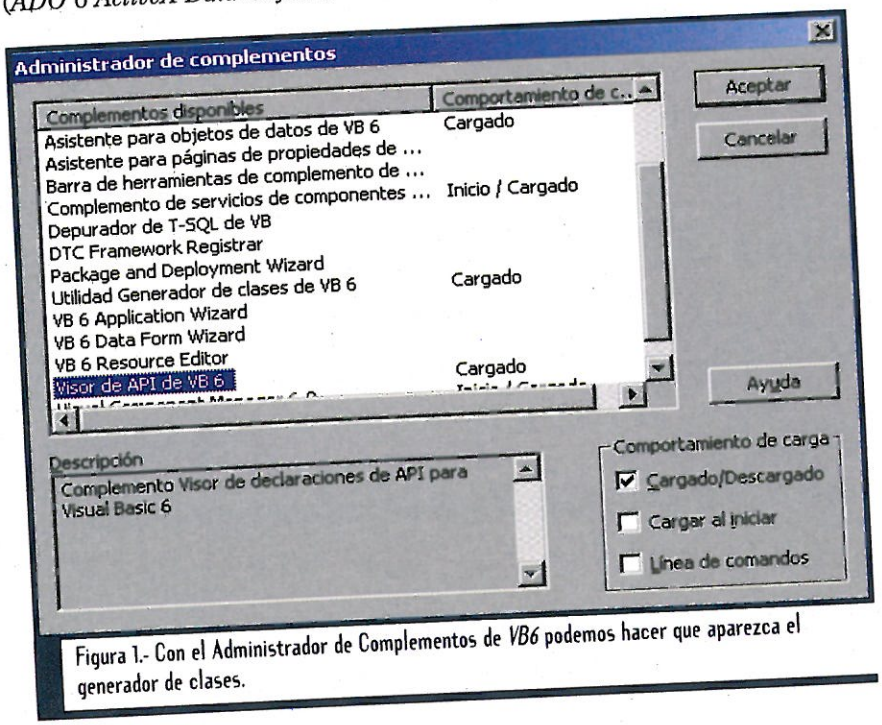


Figura 1.- Con el Administrador de Complementos de *VB6* podemos hacer que aparezca el generador de clases.

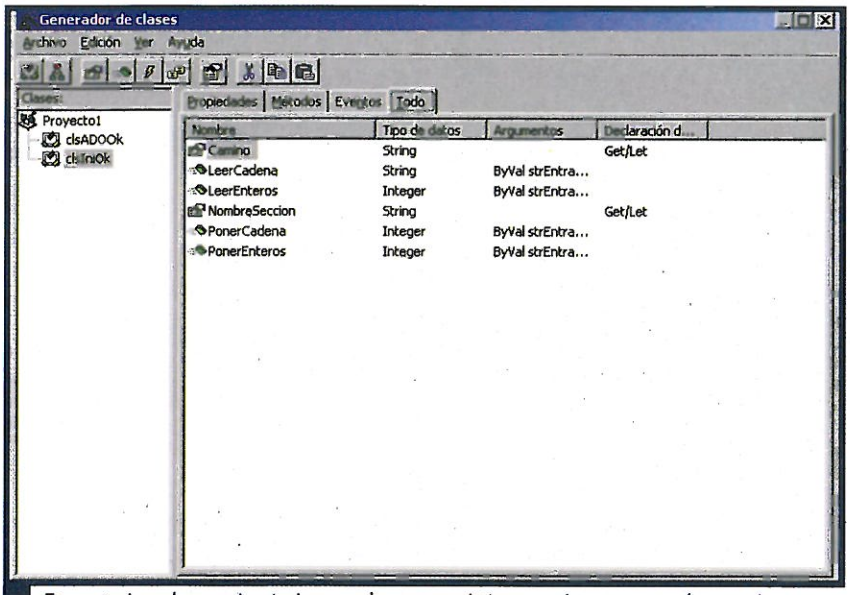


Figura 2.- Las clases, además de manualmente, también se pueden crear con el generador.

LAS PROPIEDADES

Las propiedades que vamos a implementar en esta clase de ejemplo van a ser dos: *CadenaConexion*, que indicará la cadena necesaria para conectarse a la base de datos y *mstrUltimoError*, que contendrá información sobre el último error, en caso de que haya ocurrido. Para poder utilizar las propiedades, primero las declararemos en la sección de declaraciones del módulo de clase del siguiente modo:

```
Private m_sCadenaConexion As
    String
Private mstrUltimoError As String
```

Para usar las clases de objetos ADO deben estar activadas sus referencias

Y después le asignaremos los procedimientos correspondientes para que las propiedades realicen su función, es decir, *Get* suponiendo que la propiedad sea de solo lectura, *Let* suponiendo que sea de sólo escritura, o ambos, en el caso de que la propiedad sea de lectura y escritura.

Las propiedades deberán tener el procedimiento Get o Let, según sean de lectura o de escritura

En el ejemplo de la propiedad *CadenaConexión* utilizada en algunos métodos para indicar al programa la cadena de conexión de la base de datos, veremos, tal y como hemos explicado en anteriores artículos, cómo se establece la variable miembro declarada dentro de la clase (procedimiento *Let*) o bien se devuelve el valor de dicha variable miembro (procedimiento *Get*):

```
Public Property Get CadenaCone-
    xion() As String
    CadenaConexion = m_sCadenaCo-
    nexion
End Property

Public Property Let CadenaCone-
    xion(ByVal NewValue As
    String)
    m_sCadenaConexion = NewValue
End Property
```

LOS MÉTODOS DE LA CLASE

Aquí implementaremos tres métodos: *RECVacio*, que servirá para saber si un determinado objeto *recordset* se encuentra vacío o no, *Ejecutar*, que ejecutará una sentencia *SQL (Structured Query Language)* en la base de datos, y *NumeroFilas*, que devolverá el número de filas de una tabla, pudiendo filtrarla a través de una sentencia *WHERE* del lenguaje *SQL*.

En el método *RECVacio* enviamos un *recordset* como argumento, de manera que nos devolverá un valor *booleano* que indica en todo momento si el *recordset* se encuentra vacío o no. Después comprobamos si el argumento enviado es realmente un objeto, y después comprobamos que la posición incial es igual a la final, lo que indicará que se encuentra vacío. En este caso hacemos que la función devuelve un valor verdadero, y falso en el caso contrario.

```
Public Function RECVacio(ByVal
    REC As ADODB.Recordset)
    As Boolean

    On Error GoTo ErrEtiqueta

    RECVacio = True
    If Not REC Is Nothing Then
        If REC.BOF = True And
    REC.EOF = True Then
            RECVacio = True
        Else
            RECVacio = False
        End If
    End If
    Exit Function

ErrEtiqueta:
    mstrUltimoError = Err.Number
    & " - " & Err.Description
```

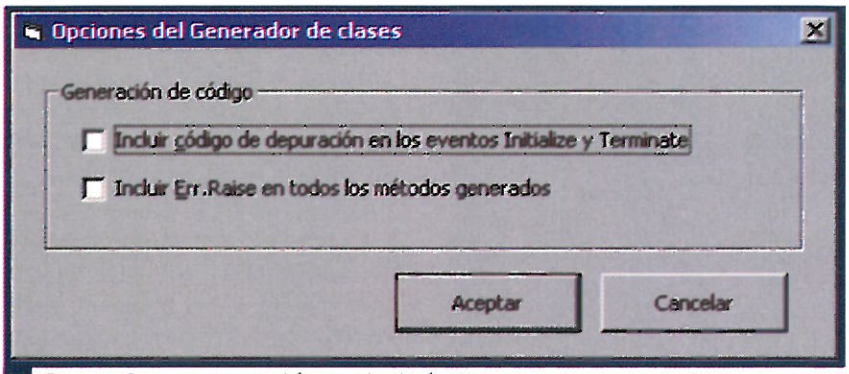

Opciones del Generador de clases
Generación de código
Incluir código de depuración en los eventos Initialize y Terminate
Incluir Err.Raise en todos los métodos generados
Aceptar
Cancelar

Figura 3.- Diversas opciones del generador de clases.

```
    RECVacio = True

End Function
```

Dentro del código, y suponiendo que ya hubiésemos establecido la propiedad *CadenaConexion*, y que la variable que referencia la base de datos se denominase *db*, podríamos incluir el siguiente código para probar el método:

```
Dim clsADO As clsADOOk
Dim strsql As String
Dim recTmp As Recordset
Dim booEstado As Boolean

Set recTmp =
  db.OpenRecordset("SELECT * FROM
    TablaPrueba")

booEstado = clsADO.RECVacio
    (recTmp)
If booEstado = True Then
    MsgBox "El recordset está
    vacio"
End If
recTmp.Close
```

Otro método que implementaremos para ejecutar sentencias *SQL* directamente en el motor de la base de datos, es *Ejecutar*.

En éste, lo primero que hacemos es establecer un control de errores, posteriormente declaramos una variable con la conexión (variable *conTmp*), y utilizando dicha variable, abrimos la conexión a la base de datos. A continuación, abrimos una transacción (*.Open*) , ejecutamos la sentencia *SQL* (*.Execute*), cerramos la transacción (*.CommitTrans*) y cerramos la conexión (*.Close*). Seguidamente destruiremos la variable y saldremos de la función. En caso de error, en primer lugar notificaremos su número con la variable miembro *mstrUltimoError* a la propiedad *UltimoError*. Después miraremos si la conexión ha quedado abierta, en cuyo caso la cerraremos y desharemos la transacción.

```
Public Function Ejecutar(SQL As
    String) As Boolean

    On Error GoTo ErrEjecutar

    Dim conTmp As New ADODB.Con-
    nection
    With conTmp
        .ConnectionString = Cade-
    naConexion
        .CursorLocation = adUse-
    Server
        .Open
        .BeginTrans
        .Execute SQL
        .CommitTrans
        .Close
    End With
    Set conTmp = Nothing
    Ejecutar = True
    Exit Function

ErrEjecutar:
    mstrUltimoError = Err.Number
    & " - " & Err.Description
    If conTmp.State = adStateOpen
    Then
        conTmp.RollbackTrans
        conTmp.Close
    End If
    Set conTmp = Nothing
    Ejecutar = False

End Function
```

Para probar este método, y siempre suponiendo que la propiedad *CadenaConexion* tiene un valor válido, ejecutaríamos el siguiente código:

```
Dim clsADO As clsADOOk
Dim booEjecutado As Boolean

booEjecutado = clsADO.Ejecutar
    ("DELETE * FROM TablaPrueba")
```

Con el método Ejecutar podremos ejecutar sentencias directamente en el motor de la base de datos

De manera similar, veremos el método *NumeroFilas,* que retornará un número (de tipo *Long*) que será el número de filas que contiene el *recorset*. Junto con el nombre del método de la clase, también le podemos proporcionar a la función dos argumentos: el primero será la tabla en la que se debe contar el número de registros, y el segundo, opcional, será una sentencia condicional *WHERE*, que delimitará un subconjunto dentro de la tabla que examinar.

La estructura se asemeja al método visto anteriormente, ya que lo primero que encontraremos es la típica sentencia de control de errores. Después, declaremos las variables que vamos a utilizar, y iniciamos el número de filas leídas a *0*. Seguidamente, comprobamos si existe el argumento opcional o no

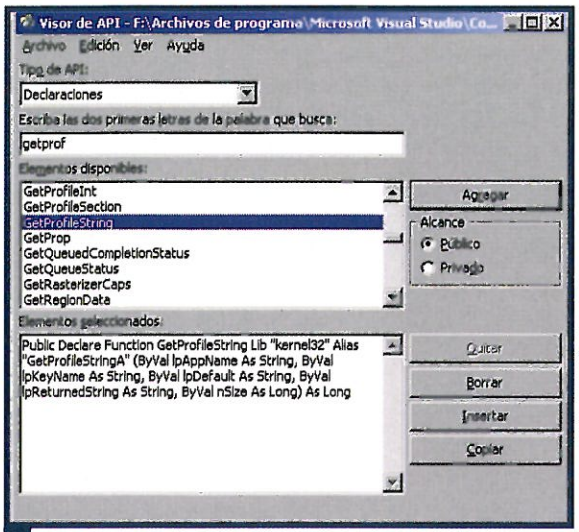


Figura 4.- Con el visor de *API* obtendremos las declaraciones de las funciones de la *API* de *Windows*.

(en cuyo caso la variable *strCondicion* contendrá una cadena vacía) y según el caso, construiremos una sentencia *SQL* de selección sobre toda o parte de la tabla. Preparada la sentencia, sólo queda ejecutarla y recuperar el valor del *recordset* (que en este caso contendrá un sólo registro con el valor del número de filas que ha encontrado la sentencia *SQL*). En la siguiente línea destruimos la variable del *recordset*, para evitar que utilice memoria innecesariamente y salimos de la función.

```
Public Function NumeroFilas(str-
    Tabla As String,
       Optional strCondicion As
    String = "") As Long

    On Error GoTo LocalError

    Dim recTmp  As New
    ADODB.Recordset
    Dim strSQL As String
    NumeroFilas = 0
    If strCondicion <> "" Then
        strSQL = "SELECT COUNT
    (*) FROM " & strTabla &
            " WHERE " & strCon-
    dicion
    Else
        strSQL = "SELECT COUNT
    (*) FROM " & strTabla
    End If
    With recTmp
        .ActiveCon-
nection = CadenaCone-
xion
        .CursorLoca-
tion = adUseClient
        .LockType =
adLockReadOnly
        .CursorType
= adOpenKeyset
        .Source =
strSQL
        .Open
        Set .Active-
Connection = Nothing
    End With
    NumeroFilas =
    recTmp.Fields(0).Value
    Set recTmp = Nothing
    Exit Function

LocalError:
    mstrUltimoError = Err.Number
    & " - " & Err.Description
    If recTmp.State = adStateOpen
    Then
        recTmp.Close
    End If
    Set recTmp = Nothing
    NumeroFilas = -1

End Function
```

Observaremos que con la clase construida, aumenta la simplicidad de nuestro programa, ya que sin el uso de las clases debemos limitarnos al uso de mucho código redundante.

Con la clase ADO simplificamos nuestro programa, al evitar código redundante

También incrementaremos la reutilización de nuestro código, ya que al empezar un nuevo proyecto, ya dispondremos de muchas de las rutinas base que lo conforman.

UNA CLASE PARA FICHEROS DE INICIO

Otro ejemplo que vamos a ver, será una clase que sirva para trabajar con ficheros de inicio, los famosos ficheros *.ini*. Aunque actualmente *Microsoft* usa el registro para almacenar variables de configuración, la verdad es que el uso de ficheros de inicio sigue siendo perfectamente válido.

DECLARACIONES DE LA API

Lo primero que deberemos hacer para leer o escribir en los ficheros de inicio, es declarar las funciones de la *API (Aplication Programming Interface)*, que son como las librerías propias del sistema operativo.

Dentro del módulo *Kernel32* encontraremos las librerías que nos interesan y que en este caso son las siguientes (la declaración completa se puede ver mediante el visor de *API*, o bien examinando el código fuente):

- *GetPrivateProfileInt*
- *GetPrivateProfileString*
- *GetPrivateProfileSection*
- *WritePrivateProfileString*

LAS PROPIEDADES

Después de las declaraciones, debemos declarar las variables miembro que harán la función de propiedades. Lo haremos en la sección de declaraciones del módulo de clase:

```
Private mstrNombreSeccion As
    String
Private mstrCaminoFichero As
    String
Private mbooInicializado As Boo-
    lean
```

Examinemos más de cerca el código de la asignación de nombre de sección. Lo primero que haremos es asignar a nuestra variable interna el valor que nos ha enviado la función. Inmediatamente comprobamos si la variable tiene un valor distinto de la cadena vacía, caso que indicará que se ha establecido el nombre de sección. Como consecuencia, marcaremos una variable *booleana* como cierta, y será la que utilizaremos posteriormente en otros procedimientos para comprobar si ha llegado a buen fin:

```
Public Property Let NombreSec-
    cion(ByVal strSeccion As
    String)

    mstrNombreSeccion = strSec-
    cion

    If Trim$(strSeccion) = ""
    Then
        mbooInicializado = False
    Else
        mbooInicializado = True
    End If

End Property
```

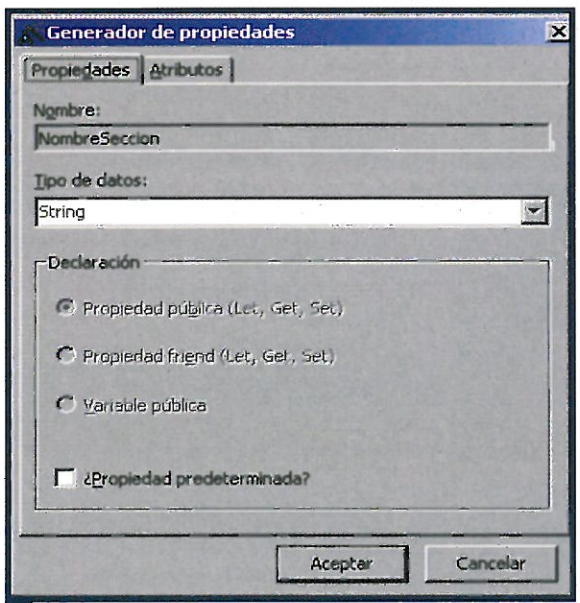


Figura 5.- Con el generador de Propiedades del Generador de Clases podremos ver y variar la propiedades de una clase dada.

Y para la función *Get*, es decir, para el procedimiento que recupera el valor de la propiedad, lo que hacemos es devolver mediante la función el valor de nuestra variable interna que apunta al nombre de sección:

```
Public Property Get NombreSec-
    cion() As String

    NombreSeccion = mstrNombre-
    Seccion

End Property
```

LOS MÉTODOS DE LA CLASE

Hemos desarrollado cuatro métodos en esta clase: uno para leer cadenas, otro para escribirlas, y sus homólogos con datos de tipo numérico. El primer método sirve para leer números de un fichero de inicio, y en él, lo primero es comprobar si previamente se ha iniciado la información de la clase, es decir, si se ha asignado valor a la sección. En caso contrario, mostraremos un mensaje de error y saldremos del error, y suponiendo que esté todo correcto invocaremos a la función *GetPrivateProfileInt* de la *API* de *Windows*:

```
Public Function Lee-
rEnteros(ByVal strEn-
trada As String,
     ByVal intValor-
Defecto As Integer)
As Integer

    If Not mbooIni-
cializado Then
      MsgBox "Mensa-
je de error"
      Exit Function
    End If

    LeerEnteros =
GetPrivateProfi-
leInt(mstrNombreSec-
cion,
        strEntrada,
intValorDefecto,
    mstrCaminoFichero)
End Function
```

Del mismo modo funcionan los métodos *PonerCadena* y *PonerEnteros*:

```
Public Function PonerCadena(ByVal
    strEntrada As String,
    ByVal StrValue As String) As
    Integer

    If Not mbooInicializado Then
      MsgBox "Error"
      Exit Function
    End If

    PonerCadena = WritePrivateP-
    rofileString(mstrNombreSec-
    cion, strEntrada, StrValue,
    mstrCaminoFichero)
End Function

Public Function
    PonerEnteros(ByVal strEntrada
    As String,
    ByVal intValor As Integer) As
    Integer

    If Not mbooInicializado Then
        MsgBox "Error"
        Exit Function
    End If

    PonerEnteros = WritePrivateP-
    rofileString(mstrNombreSec-
    cion,
          strEntrada, CStr(intVa-
    lor), mstrCaminoFichero)
End Function
```

Para acceder a los ficheros de inicio debemos declarar las funciones de la API de Windows

El método *LeerCadena* es algo diferente. En él, lo primero que haremos es declarar las variables temporales, pero la variable de tipo cadena (*strTemp*) la declararemos

como un *array* de *50* posiciones. Y después del control de errores leeremos a través de la sentencia *GetPrivateProfileString*. Hasta aquí la estructura es como en los demás métodos, la diferencia es que después de leer la cadena correspondiente, seleccionamos de ésta los valores que nos interesan (que serán iguales a la longitud de la cadena leída), ya que al tener más posiciones el *array*, éste podría contener "caracteres basura".

Hay que eliminar los caracteres sobrantes al leer una cadena de inicio para un array

```
Public Function LeerCadena(ByVal
    strEntrada As String,
    ByVal strCadenaDefecto As
    String) As String

    Dim strTemp As String * 50
    Dim lngRetVal As Long

    If Not mbooInicializado Then
      MsgBox "Error"
      Exit Function
    End If

    lngRetVal = GetPrivateProfi-
    leString(mstrNombreSeccion,
    strEntrada,
        strCadenaDefecto, str-
    Temp, Len(strTemp), mstrCami-
    noFichero)

    If lngRetVal Then
      LeerCadena = Left$(strTemp,
    lngRetVal)
    End If
End Function
```

PROBANDO LA CLASE

Suponiendo que ya tenemos la clase de los ficheros de inicio incorporada en nuestro programa, ¿qué tendremos que hacer para utilizar sus funcionalidades? En primer lugar debemos declarar una variable que apunte a la clase citada (el ámbito de la variable depende de los sitios donde utilicemos la clase). En nuestro caso, lo haremos en la sección de declaraciones del formulario principal:

```
Dim clsINI As New
    clsIniOk
```

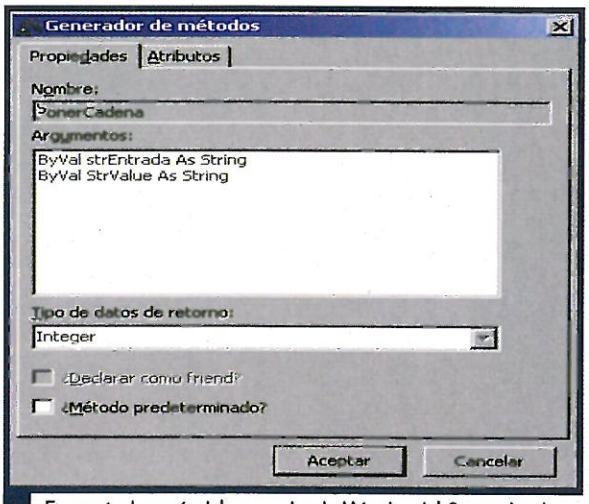


Figura 6.- A través del generador de Métodos del Generador de Clases podemos ver y variar los métodos de una clase dada.

Conociendo que nuestra variable se llama *clsINI* vamos a suponer ahora que deseamos leer el contenido de un fichero de inicio. Entonces declararemos dos variables para almacenar los valores temporalmente: *Dim strTmp As String* y *Dim intTmp As Integer*. Establecemos la de nuestro fichero (debemos ir con cuidado, ya que al ejecutar el código fuente puede ser que de un error): *clsINI.Camino = "D:\sp\prueba.ini"* y nombramos la sección que queremos leer: *clsINI.NombreSeccion = "prueba1"*

Usar el registro para almacenar variables de configuración es perfectamente válido

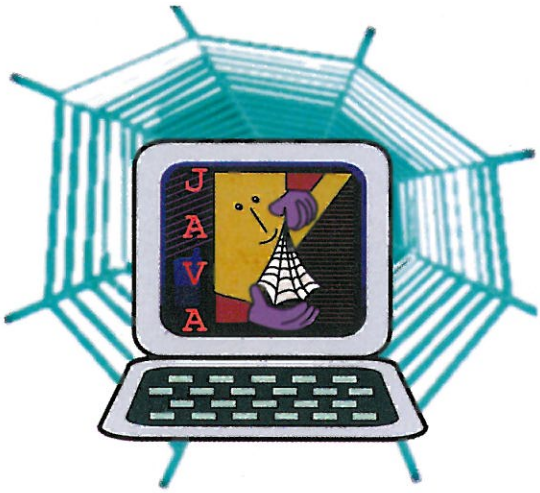
Y ahora leeremos una cadena, pasando como argumento al método de la clase la línea que queremos leer, y teniendo en cuenta la cadena que se devolverá por defecto en caso de que no se pueda leer. Seguidamente cambiamos de sección y leemos un entero de la segunda sección, para posteriormente, mostrar dos mensajes en los que se verán los valores:

```
strTmp =
    clsINI.LeerCadena("linea1",
    "Error")
clsINI.NombreSeccion = "prueba2"
intTmp =
    clsINI.LeerEnteros("linea2",
    0)
MsgBox strTmp
MsgBox intTmp
```

Si lo que queremos es escribir valores, el código será muy similar al anterior, con la salvedad del cambio de los métodos invocados en la clase y de sus correspondientes argumentos.

```
intTmp = 0
intTmp =
    clsINI.PonerEnteros("linea2",
    1)
clsINI.NombreSeccion = "prueba1"
strTmp =
    clsINI.PonerCadena("linea1",
    "Valor Cambiado")
MsgBox intTmp
MsgBox strTmp
```

Concluimos con estos ejemplos prácticos la aproximación a la *POO* en *Visual Basic*, que si bien, tal y como hemos visto no es una *POO* pura, sí que va a servir para dar una mayor claridad a nuestro código y una mayor reutilización del mismo.



Mejora del rendimiento de aplicaciones Java

Optimize It contra JProbe

Javier Sanz Alamillo. *Ingeniero de Software.*

El proceso de mejora de rendimiento de una aplicación puede presentar diferentes problemas, desde los clásicos, en los que se necesita modificar el código, hasta los más complicados, como los *loiterers*, que requerirán el uso de herramientas de análisis específicas.

Para mejorar nuestra aplicación, disponemos de una serie de herramientas que nos informan sobre los métodos más utilizados, cantidad de objetos creados, tamaño del heap, si éste es más o menos estable, etcétera. El estudio de todos estos parámetros se basa en dos tipos de analizadores, llamados *performance profilers* y *memory checkers/memory profiling/memory debuggers.*

Los *performance profilers* o analizadores de rendimiento ofrecen un gran conjunto de información referida al tiempo de ejecución, como por ejemplo, qué tanto por cierto de ejecución consume cada método, cuántas llamadas han ejecutado, si alguno produce un cuello de botella, etcétera, con lo que el desarrollador puede investigar si todo es correcto y valorar la conveniencia de una optimación.

Los programas del tipo *memory profiling* o analizadores de comportamiento de memoria (también se les denomina de *tuning,)* están dirigidos a ofrecer la máxima información sobre cómo se está gestionando la memoria, qué objetos se crean más veces, cómo se están destruyendo, las actuaciones del recolector de basura, etcétera.

Todo un conjunto de datos para evitar los *loiterers* y conseguir una aplicación más estable, pudiéndose mejorar, por ejemplo, la creación de objetos, como se describió anteriormente.

En la actualidad, existe en el mercado una amplia variedad de productos que ofrecen todas éstas

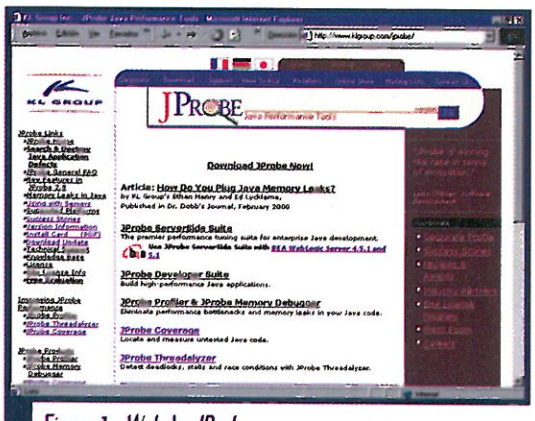


Figura 1.- Web de JProbe.

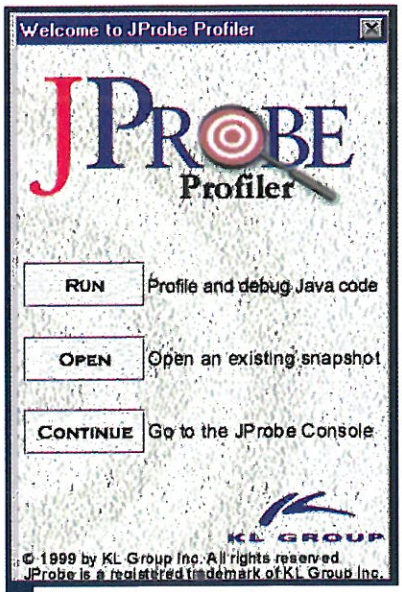


Figura 2.- Opciones de *JProbe*.

posibilidades. A continuación hablaremos de dos de los programas más utilizados para la mejora en el rendimiento de las aplicaciones, tanto en velocidad como en estabilidad, *OptimizeIt,* de *Intuituve Systems,* y *Jprobe,* de *KLG Group*. Sólo mencionaremos ahora la existencia de otro producto, *TrueTime* de *Numega,* que aunque no está suficientemente maduro, sigue abriéndose camino poco a poco, y habrá que tenerlo en cuenta de aquí a poco tiempo.

Se trata de dos productos comerciales, aunque el lector podrá realizar unos mínimos pero valiosos análisis de velocidad con la máquina virtual del *JDK 1.2* que incluye una opción, *-Xrunhprof* y muestra entre otros datos, las veces y el tiempo de ejecución de los métodos. A continuación, con el fichero generado, puede usar *HAP 1.0.3* (*Heap Analisis Tool*) y estudiar los resultados obtenidos. Esta última herramienta se puede descargar de *Sun* para *Java*, en http://java.sun.com .

También hay disponibles algunas herramientas gratuitas fuera del entorno corporativo, como *HyperProf*, que desafortunadamente está empezando a pasar al plano comercial. Como memory *profiling* gratuito puede buscar *JUM*, basado en *JVMPI* (*Java Virtual Machine Profiling Interface*) de *Java*.

Los análisis de rendimiento son muy importantes.

Empezaremos con la presentación de *JProbe*, y posteriormente estudiaremos *OptimizeIt,* para terminar con una comparativa de ambos, con lo que el desarrollador podrá seleccionar el que mejor se adapte a sus necesidades.

JPROBE

JProbe 2.5 es uno de los productos más utilizados para el análisis de grandes aplicaciones *Java.* Permite responder a preguntas como:

- ¿Cuáles son los métodos, que más que ejecutan, y cuánto más?
- ¿Cuáles son las líneas de mi código que se ejecutan con más frecuencia, y cuales consumen más recursos, tanto en tiempo *CPU* como en memoria?
- ¿Puedo comparar la mejora en tiempo tras una optimización?
- ¿Cómo puedo encontrar los cuellos de botella? y, ¿ puedo llegar a sospechar si está en mi programa, en la máquina virtual o incluso en mi sistema operativo?

Todas esas preguntas se pueden contestar tras un análisis de la aplicación con este programa.

Mediante *JProfiler* y *Memory Debugger* (todo integrado en *JProbe*) se pueden mejorar las aplicaciones en *Java 2* reduciendo entre otros, posibles cuellos de botella y una excesiva creación de objetos.

INSTALACIÓN

La instalación de *JProbe* es una tarea que se basa en el seguimiento de las instrucciones que va mostrando el asistente dispuesto al efecto, y tras seleccionar unas cuántas opciones, podremos empezar a trabajar con el nuevo producto.

Antes de detallar los requerimientos, indicaremos que *JProbe* está disponible para *Windows 95, 98, NT, 2000* y *Solaris*. Para utilizar con *Windows 2000* hay que aplicar un *patch* disponible en el *Web* de *KLG*.

Básicamente, los requerimientos *hardware* necesarios para el correcto funcionamiento de *JProbe* en una plataforma *PC/Windows* son:

- El procesador mínimo requerido no debería de ser nada inferior a un *Pentium* 200 *Mhz*. Tras algu-

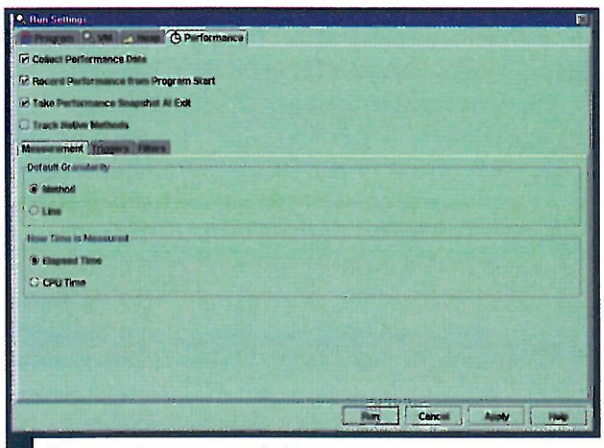


Figura 3.- *Run Setting* de *Probe*.

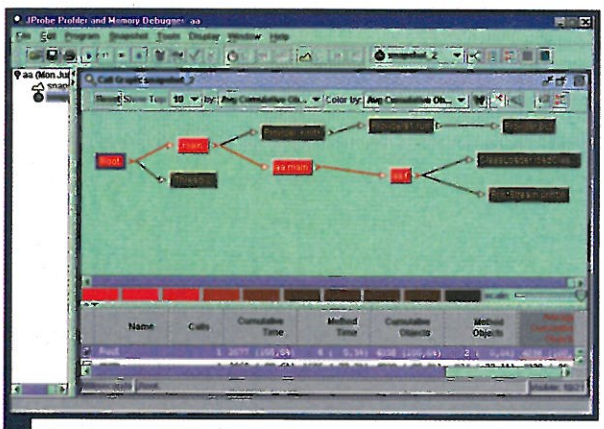


Figura 4.- *Grafo de ejecución.*

nas pruebas, el uso todo lo inferior a un *Pentium* II a 300 *Mhz* se queda algo justo, por lo que es muy recomendable trabajar con 400 *Mhz*. Un *Athlon* o un *Pentium III* hacen que el producto parezca unaaplicación estilo *Windows* normal.

- La cantidad de *RAM* que se requiere es de 128 *MB*. Se debería seguir la recomendación fielmente, puesto que con menos no se puede utilizar cómodamente. Se pueden recomendar 256 *MB* para una ejecución muy robusta y sin problemas.

- En vídeo, se recomienda una *SVGA*, de resolución mínima 800x600, aunque se especifica que lo ideal sería 1024x768.

Los requerimientos *hardware* no son muy elevados, si lo comparamos con los equipos actualmente disponibles para la mayoría de los proyectos.

El tamaño de disco necesario para su instalación no supondrá ningún problema, ocupará unos 30 *MB* aproximadamente.

CARACTERÍSTICAS GENERALES

Mediante el uso de la versión *2.5* de *JProbe*, que integra a *JProbe Profiler* y *Memory Debugger*, se pueden realizar análisis de rendimiento de las aplicaciones *Java* de la forma más rápida y sencilla, consiguiéndose mejoras muy interesantes en las aplicaciones.

Con el uso del *profiler* se detectan cuellos de botella causados por el uso de algoritmos poco eficientes, excesiva creación de objetos, bloqueos de entrada o salida, etcétera.

La posibilidad de un doble tipo de análisis ayuda en la búsqueda de loiterers.

Gracias a la interface gráfica, se dispone de una serie de grafos e información sobre los objetos, que permite ver el tiempo de ejecución de cada línea de nuestro código.

El uso de *Memory Debugger* permite estudiar los posibles *loiterers*, siguiendo el crecimiento de un determinado tipo de objeto, viendo el total de objetos creados, el movimiento del recolector de basura, cómo ha ido evolucionado el *heap* de la aplicación, etcétera, todo ello visualmente, facilitándose así ésta difícil tarea.

Específicamente, mediante el *Profiler* obtenemos información del tiempo acumulado de ejecución, total de objetos creados por tipo, total de llamadas a métodos, tiempo medio de ejecución, tiempo acumulado por método, objetos por método...

Mediante *Memory Debugger* tenemos un control sobre la memoria utilizada, la relación entre los objetos y métodos que los unen, estados del *heap*, o acceso al código fuente. Se puede gestionar el recolector de basura para ver el impacto de su ejecución sobre nuestro programa.

En la versión *2.5* se han incluido las siguientes mejoras:

- Soporte propio para los *JDK* versiones *1.1.7B, 1.1.8, 1.2.1* y *1.2.2,* tanto para *NT* y *Solaris*.

- Dos ediciones del producto, *ServerSide* y *Developer*, que detallaremos seguidamente.

- Soporte a *Solaris*. La edición *ServerSide* de *JProbe* está disponible para *Solaris* 2.6 y 7, utilizando *JDK* 1.1 y *JDK* 1.2.

- Adaptación con *VisualAge 2.5*, integrándose en el repositorio de éste, permitiendo así una gestión más simple y cómoda dentro del entorno.

- Mejoras en el *JIT* y en el *IDE*, siendo más intuitivo.

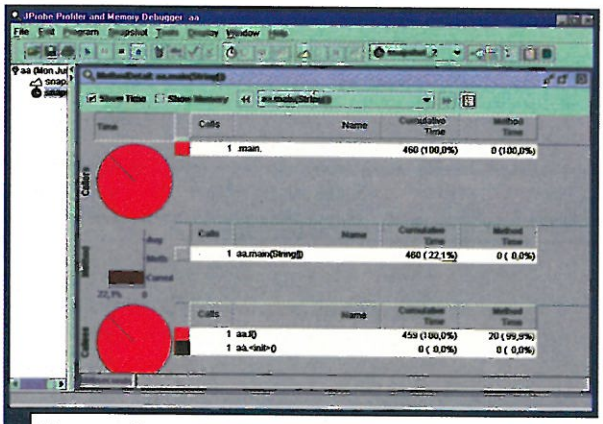


Figura 5.- *Memory Debugger.*

TABLA 1. Plataformas y versión JDK.

Edición	Windows		Solaris	
	JDK 1.1	*JDK 1.2*	*JDK 1.1*	*JDK 1.2*
Developer				
Profiler	X	X		
Memory Deb.	X	X		
ServerSide				
Profiler	X	X	X	X
Memory Deb.	X	X	X	X

Gracias a la separación entre el *profiler* y el *memory debugger*, se pueden realizar estas tareas sin que el exceso de información pueda llegar a crear confusiones.

DEVELOPER Y SERVERSIDE

La edición *Developer* está indicada aplicaciones de estilo tradicional y la creación de proyectos en el lado cliente. Se puede decir que es como una edición estándar, atendiendo a la forma en la que se suelen clasificar los entornos visuales para *Java*.

Con todo el potencial de la edición *ServerSide*, por su matiz de lado servidor, permite un análisis muy completo en las clásicas aplicaciones que no exigen de un rendimiento elevado o de un control minucioso de su comportamiento.

La edición *ServerSide* es la más completa, al estar preparada para realizar los estudios de aplicaciones de alto rendimiento en el lado servidor. Permite estudiar servidores que soportan elevadas cargas de trabajo, localizando rápidamente los clásicos cuellos de botella, y la creación de objetos y *loiterers*.

Se puede utilizar en *Servlets*, *JSP* (*JavaServer Page*), lo que hace que su integración con las aplicaciones *Web* sea muy sencilla. En esta edición se añaden las siguientes características:

- Ejecución en *Solaris*.
- La utilidad *Server Launch Pad*, que permite su integración con los servidores de aplicaciones más utilizados.
- Funcionamiento en local y remoto.
- Uso especial a través de la línea de comandos, evitando así sobrecarga de información de la parte de grafo visual.
- Integración con *IBM VisualAge 2.5*.

PLATAFORMAS

En la Tabla 1 se muestra una relación entre la plataforma de uso y la versión de *JDK* que se puede aplicar.

Recomendaremos al lector que, salvo utilización del *JDK 1.2*, no debería usar un *JDK* inferior que no fuera el que forma parte del producto, ya que se han detectado ciertas incompatibilidades.

Para utilizar *JProbe* en *Windows 2000*, hay que aplicar un parche. La versión para *Solaris* se ejecuta sólo en *UltraSPARC*, no en *SPARC Station* o *Intel Solaris*. El *profiling* remoto entre *Windows* y *Solaris* requiere de *Samba* y *NFS*.

CONCLUSIONES

Se puede decir que *JProbe 2.5* ofrece un conjunto muy completo de posibilidades que facilita al desarrollador analizar sus programas para mejorar el rendimiento y la estabilidad.

Gracias a la completa información -a veces, incluso demasiada, y hay que eliminar determinadas referencias-, se puede apreciar partes del código concretas que no son accesibles, y detectar que algo no marcha del todo bien.

JProbe ofrece toda la información necesaria para mejorar el rendimiento de los desarrollos

Su velocidad de ejecución es muy elevada, aunque cuando la aplicación es algo voluminosa, con muchas clases, le cuesta seguir y es preferible deshabilitar alguna de las posibilidades de gestión del *heap* y del análisis de tiempos, como por ejemplo, eliminar el control de objetos que no son parte de nuestro programa.

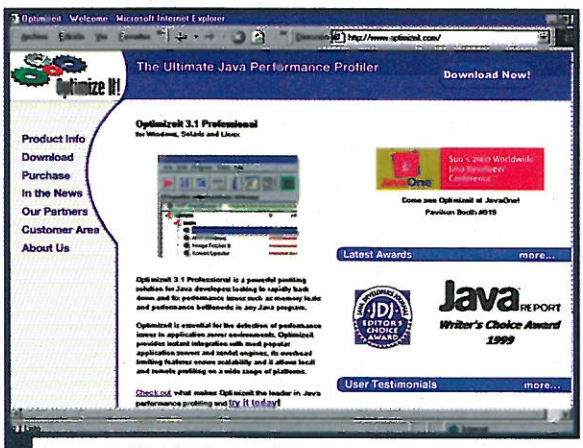


Figura 6.- *Web de OptimizeIt.*

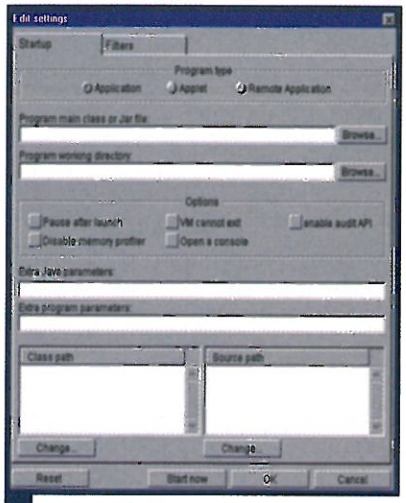


Figura 7.- Configuración inicial.

Se ha mejorado la respuesta al análisis de aplicaciones que utilizan *Swing*, ya que éstas soportan la mayor carga de objetos y *CPU*. De este modo, se puede mejorar la búsqueda de *loiterers* típicos de *Swing*.

OptimizeIt se puede integrar en la mayoría de los entornos Java.

Si bien el conjunto de información que se presenta es correcto y razonable, a veces es algo engorroso y difícil de manejar. El entorno visual ha mejorado, aunque en algunos momentos todo se llena de ventanas y cuesta buscar algún dato específico.

Para terminar, indicamos que los precios para *JProbe 2.5* para la edición *Developer* para *Windows* es de 630_ (105.000 Pts.), y sólo 131 _ (21.800 Pts.) si se actualiza de la versión *2* a la *2.5*.

En la edición *ServerSuite* tenemos que desembolsar 788 _, y para pasar de la versión *2* a la *2.5* ,unos 262 _. Para *Solaris*, tenemos únicamente la edición *ServerSuite*, al precio de 788 _ (131.000 Pts.)

OPTIMIZEIT PROFESSIONAL 3.1

Intuitive Systems ofrece al mercado de los analizadores de aplicaciones el *OptimizeIt Professional 3.1*, un completo producto que permite realizar un análisis de rendimiento de las aplicaciones *Java*, gracias al amplio conjunto de información que ofrece. Gracias a él, se pueden conseguir mejoras en la estabilidad y ejecución de los desarrollos. Veamos las características del producto.

INSTALACIÓN

Consiste en el seguimiento de las instrucciones que va mostrando el asistente. Tras un periodo corto de tiempo, el producto está listo para ser utilizado.

OptimizeIt Professional 3.1 está disponible para múltiples plataformas de desarrollo, lo que le hace muy interesante, ya que permite comparar qué resultados de rendimiento ofrece una misma aplicación en diferentes plataformas, garantizando no sólo una alta portabilidad del propio desarrollo en sí, sino que demuestra la estabilidad que puede ofrecer la aplicación en diferentes entornos.

Los entornos disponibles son *Windows* 95, *98*, y *NT*. No hay referencias a *Windows 2000*, aunque es de suponer su compatibilidad, en estaciones *Sun* con *Solaris 2.6* y *2.7* y bajo *Linux*, *Red Hat* 5.2.

Los requisitos *hardware* en caso de la versión para *PC/Windows* son:

- Al igual que otros productos similares, no es recomendable nada menor a *Pentium* a 200 *Mhz*. Lo mínimo para trabajar correctamente sería un *Pentium* II, pero claro está, nada como un *Pentium III* o un *Athlon* para que todo se desarrolle como la seda.
- La cantidad d mínima recomendable de *RAM* es de 128 *MB*. Todo va bien con esa cantidad, pero en aplicaciones voluminosas, el resultado se vuelve lento y parece que pide más memoria. Se recomienda 256 *MB* para el análisis de aplicaciones voluminosas.
- Resolución gráfica, lo estándar, una *SVGA*, de resolución mínima 800x600.

El tamaño en disco necesario es de unos 10 *Mb* aproximadamente.

CARACTERÍSTICAS

Mediante el uso de *OptimizeIt 3.1*, los desarrolladores pueden probar y mejorar el rendimiento de las aplicaciones *Java*, *applets*, *servlets* y *javabeans*.

Gracias al análisis de rendimiento, se pueden resolver problEmas como la creación excesiva de obje-

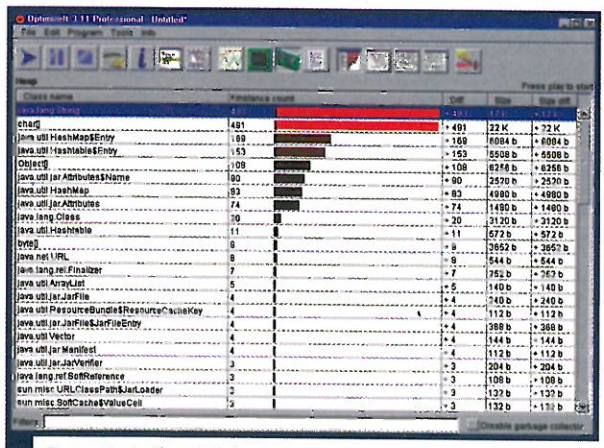


Figura 8.- Análisis del *heap*.

TABLA 2. Cuadro comparativo.

CARACTERÍSTICAS	JPROBE 2.5	OPTIMIZEIT 3.1
Entorno IDE	Sí, y línea comandos	Sí
JDK en el producto	Si, 1.1.7 a 1.2.2	No
Cantidad información del análisis y su valor	Muy completa	Completa
Presentación datos	Correcta	Muy correcta
Velocidad	Buena	Buena
Control tiempo real de GC, heap, stack	Sí	Sí
Plataformas	Windows, Solaris	Windows,Solaris, Linux
Acceso al código	Sí	Sí
Grafos referencias	Sí	Sí

tos, algoritmos ineficientes, *loiterers*, etcétera.

Mediante el denominado *OptimizeIt CPU Profiling* se obtienen todo tipo de datos como uso de *CPU* por método, el tiempo total de ejecución, tiempo de inactividad, etcétera. Gracias a estas medidas se pueden detectar los cuellos de botella más fácilmente.

Utilizando dos técnicas diferentes de análisis, se puede mejorar la medida del rendimiento, adaptándose así mejor los resultados, tanto para grandes aplicaciones que usan objetos grafos, como para aquellas que simplemente son pequeños *servlets*.

Información como cuáles son los métodos que más tiempo se han ejecutado, o cuántas veces se han invocado, se consigue rápidamente mediante *Hotspot Display*.

Se puede realizar un seguimiento total del rendimiento de la aplicación en tiempo real, pudiéndose incluir marcas de seguimiento y control.

Tampoco descuida el control de los *threads*, pudiéndose apreciar cuántos *threads* se están ejecutando, cuánto tiempo consume cada uno, si producen bloqueos, etcétera.

Además, se puede acceder en cualquier momento al código fuente para apreciar los métodos o las líneas que producen ese cuello de botella, o averiguar la razón del consumo tan elevado de *CPU*.

eliop

Si estás interesado en desarrollar con nosotros SOFTWARE en un entorno de gran CALIDAD PROFESIONAL, te ofrecemos la oportunidad de incorporarte a una empresa de ALTA TECNOLOGÍA, fuertemente asentada en el MERCADO INTERNACIONAL DE SISTEMAS DE SUPERVISIÓN Y CONTROL.

Ingenieros de Desarrollo de Software

Requisitos imprencisdibles para todos los puestos:

- Titulación Superior en Informática, Telecomunicación, Industrial o similar.
- Dominio de C y C++.
- Programación bajo Unix y/o Windows NT.
- Experiencia demostrable.
- Buen nivel de inglés.

También se valorará:

- Conocimientos de librería gráfica ILOG VIEWS.
- Oracle
- Programación en PL/SQL y ProC
- Conocimientos de IPCs (Semáforos,memorias compartidas, etc.)
- Programación con Threads.
- TCP/IP
- Visual C++

Interesados enviar Curriculum Vitae a ELIOP, S.A., Avda. de Manoteras nº 30, 28.050 Madrid.

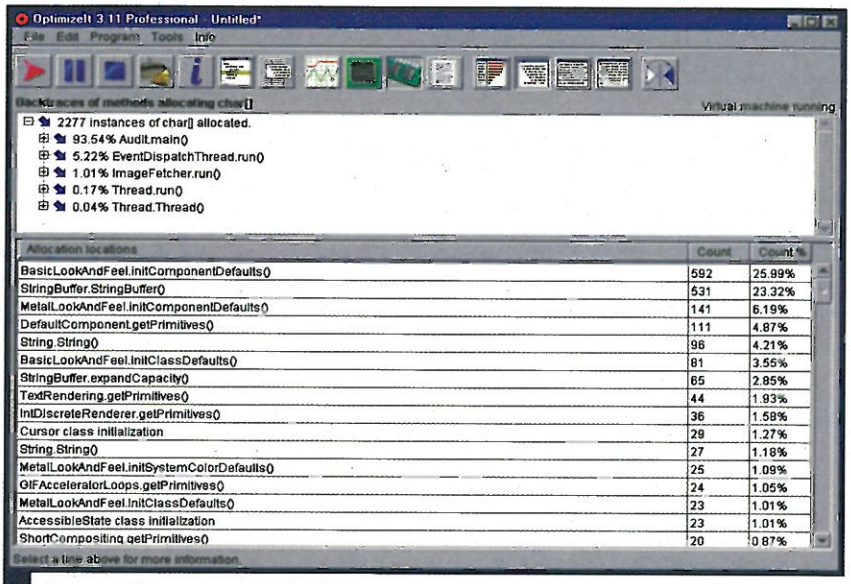

OptimizeIt 3.11 Professional - Untitled*
File Edit Program Tools Info
Backtraces of methods allocating char[]
Virtual machine running
2277 instances of char[] allocated.
93.54% Audit.main()
5.22% EventDispatchThread.run()
1.01% ImageFetcher.run()
0.17% Thread.run()
0.04% Thread.Thread()
Allocation locations | Count | Count %
BasicLookAndFeel.initComponentDefaults() | 592 | 25.99%
StringBuffer.StringBuffer() | 531 | 23.32%
MetalLookAndFeel.initComponentDefaults() | 141 | 6.19%
DefaultComponent.getPrimitives() | 111 | 4.87%
String.String() | 96 | 4.21%
BasicLookAndFeel.initClassDefaults() | 81 | 3.55%
StringBuffer.expandCapacity() | 65 | 2.85%
TextRendering.getPrimitives() | 44 | 1.93%
IntDiscreteRenderer.getPrimitives() | 36 | 1.58%
Cursor class initialization | 29 | 1.27%
String.String() | 27 | 1.18%
MetalLookAndFeel.initSystemColorDefaults() | 25 | 1.09%
GIFAcceleratorLoops.getPrimitives() | 24 | 1.05%
MetalLookAndFeel.InitClassDefaults() | 23 | 1.01%
AccessibleState class initialization | 23 | 1.01%
ShortCompositing.getPrimitives() | 20 | 0.87%
Select a line above for more information

Figura 9.- Seguimiento creación objetos.

Puede ser integrado en otros entornos de desarrollo, como *VisualAge 2.0, Symantec, JBuilder y Jdeveloper*, por lo que realizar esta tarea se convierte en algo muy cómodo.

Respecto al *memory profiling*, se puede hacer un seguimiento exhaustivo de la creación de objetos, del *heap*, detalles de clase, histograma de frecuencias, un gráfico de referencias para encontrar los posibles *loiterers*, etcétera. El conjunto de información es completo y útil.

Los movimientos en el *heap* pueden observarse en tiempo real, apreciándose incluso el crecimiento del mismo en una relación *kbs* con los objetos generados.

El recolector de basura se puede forzar manualmente y deshabilitar, viéndose así cómo se sobrecarga el *heap*.

Las nuevas características incluidas en la versión *3.1* son:

- Soporte para *Java 2*. En contra de lo que pueda parecer, es necesario configurar la máquina virtual sobre la que se va a ejecutar. No viene con ninguna preinstalada.
- Sistema dual de análisis de ejecución para una mejor adaptación a los distintos tipos de programas.
- Un nuevo grafo de referencias a objetos que ayuda a encontrar los *loiterers* y eliminarlos.
- Visualización de referencias de monitores, clases estáticas, *JNI* y el *stack* nativo.
- Integración sin problemas en aplicaciones que utilizan los entornos de servidores de aplicaciones más comunes, como *WebLogic* de *BEA, iPlanet Web Server4.0*, con motores de *Servlets*, como *ServletRun* y *Allaire JRun*.

PLATAFORMAS

Las plataformas soportadas, como se indicó anteriormente son *Windows, Solaris* y *Linux*. Debido a que para utilizar *OptimizeIt* hay que configurar manualmente la máquina virtual, no se recomienda utilizar una máquina virtual que no soporte *Java 2*, porque pueden existir características que funcionen u ofrezcan datos poco válidos.

CONCLUSIONES

OptimizedIt se muestra como un producto completo para la mejora de las aplicaciones *Java*. Permite obtener un amplísimo conjunto de información, desde el número de concreciones generadas, hasta los porcentajes de uso por *threads* en el programa.

Todo ese conjunto de información está bien presentada, y es fácilmente accesible, lo que hace que detectar determinado tipos de errores sea algo relativamente sencillo.

Su velocidad de ejecución es correcta, aunque en aplicaciones con *Swing* y de cierto volumen, ya empieza a notarse que todo no va tan rápido como uno esta acostumbrado.

La información en tiempo real es fundamental.

Para acabar, indicamos que el precio para *OptimizedIt Professional 3.1* para la plataforma *Windows* es de 525 € (87.000 Pts.) La actualización de la versión *2.0* a la *3.1* vale 104 € (17.300 Pts.)

Para *Linux*, se dispone de la versión *3.0*, que cuesta también 525 _.

En *Solaris* se dispone de la versión *3.1* y su precio es idéntico al de los anteriores. Para actualizarse desde la *2.x* a la *3.1*, hay que pagar 235 €.

COMPARATIVA

Como se puede comprobar, los dos productos son muy completos y ofrecen similares prestaciones, y ambos ofrecen muchísima informa-

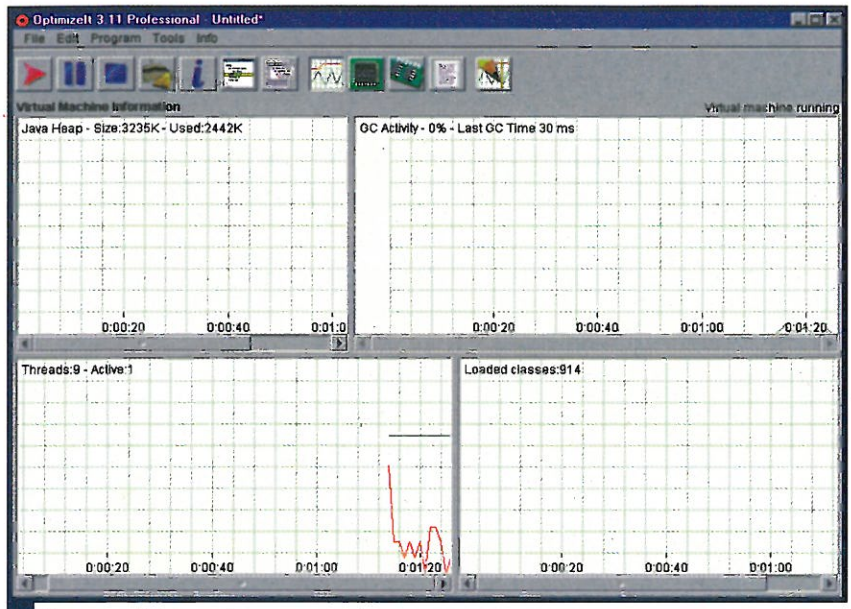


Figura 10.- Información en tiempo real.

ción y amplias posibilidades, lo que hace que decidirse por uno u otro, sea complicado.

La velocidad de ejecución de los dos productos es similar, aunque cuando se utilizan aplicaciones con *Swing, OptimizedIt* parece algo más rápido.

Los dos productos son sumamente completos

La información sobre la creación de objetos, el *heap* y la posibilidad de gestión del recolector de basura está disponible en ambos.

En la cuestión de las plataformas de ejecución, *OptimizeIt* sale victorioso, porque se introduce en un mercado que esta comenzando a tener mucha presencia, el mercado de productos para *Linux*, lo que hará que muchos desarrolladores de aplicaciones para *Internet* que utilicen servidores *Web* con motores de *Servlets* en máquinas con *Linux* se decidan por él.

El tratamiento y control sobre los *threads* para la búsqueda de interbloqueos y problemas de ejecución está correctamente resuelto en ambos.

La presentación de la información es aceptable, pero muchas veces es algo tedioso eliminar información que no es práctica para centrarse en la referente de nuestra aplicación.

JProbe presenta una mejora respecto a *OptimizeIt*, ya que integra máquinas virtuales propias, lo que implica que no haga falta una configuración previa. Claro que visto desde otro punto de vista, esto puede ser un problema, porque al no usarse la misma máquina virtual en el análisis de rendimiento que en la de producción, quizás surjan dudas sobre si se obtendrán los mismos resultados, desde luego, es de esperar que más o menos sea así.

Ambos permiten el análisis de *applets* y *servlets*, lo que facilita el estudio de servidores con una carga importante.

La detección de algoritmos ineficientes, localizándose rápidamente que método y que parte del mismo está provocando un cuello de botella, hace que no haya excusa para no modificar el código fuente y mejorar la ejecución.

En la siguiente Tabla 2 podemos ver las características más importantes de los dos productos, lo que ayudará al lector a comparar mejor las características de uno y otro.

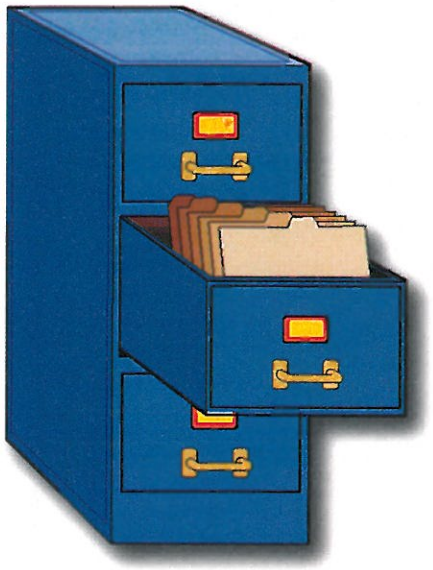
CONCLUSIONES

Gracias a los productos presentados, se pueden mejorar la estabilidad y rendimiento de una aplicación *Java* de forma cómoda y sencilla, aunque con un cierto trabajo añadido.

La estabilidad de la aplicación frente a los *loiterers* se ha presentado como un problema que tener en cuenta cuando se desarrollan aplicaciones de alto rendimiento o servidores que sufren mucha carga. Tras la explicación de la raíz del problema, se pueden adoptar políticas que minimicen su impacto y utilizando uno de los productos descritos, localizar el exceso de creación de objetos y aplicar una solución.

Los dos productos presentados son ciertamente completos. Plataformas similares, gran conjunto de información, *Java* 2, gestión del recolector de basura, control del *heap*, análisis de todo tipo, estadísticas varias, etcétera. Los requerimientos más importantes están presentes, por lo que la cuestión económica es un factor importante.

Esperamos que este artículo os haya ayudado en vuestra elección. En otros artículos hablaremos de otras mejoras y optimaciones más elementales que nos ayudarán a hacer nuestra labor más eficiente.



Aplicaciones de bases de datos con Delphi 5 (VI)

Juan Luis Ceada Ramos.
Ingeniero Superior de Informática.

La versión 5 de *Delphi* permite crear aplicaciones de bases de datos sin necesidad de usar el *BDE*, mediante los componentes *ADO* e *IB Express*. Esta característica resulta de gran utilidad en el momento de distribuir cualquier desarrollo que requiera bases de datos.

ALTERNATIVAS

Hasta la versión *4* de *Delphi*, si queríamos realizar una aplicación de bases de datos, nos veíamos obligados a utilizar el *BDE* (en cualquiera de sus versiones). El *BDE* de *Borland* es extremadamente potente, pero hay casos en los que es necesario tener en cuenta otras alternativas. Si normalmente sólo trabajamos con *Interbase*, por ejemplo, no tiene sentido cargar con un motor de bases de datos "genérico". Seria mejor, desde el punto de vista de la eficiencia temporal, usar componentes de acceso a datos específicos para este *SGBD*.

De igual forma, cuando desarrollamos pequeñas aplicaciones, cargar con el *BDE* (del que sólo usamos parte de su potencia real) puede hacer que nuestros programas no consigan el éxito esperado. Por ejemplo, ¿quién estaría dispuesto a utilizar una pequeña agenda telefónica, que con el *BDE* ocupe 15 *Mb*?

Como hemos dicho, hasta la versión *4* no había más remedio que trabajar con el *BDE*. Es cierto que hay varios sustitutos en el mercado, pero son de terceras empresas, y a nadie le gusta tener que soltar más dinero después de haber invertido 400.000 pesetas en una licencia *de Delphi C/S*. En esta

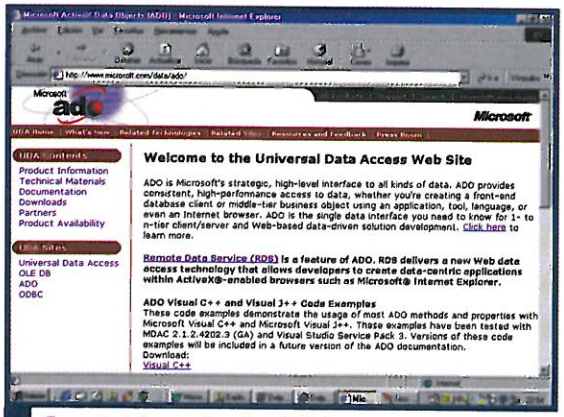

Microsoft ado
Microsoft
Welcome to the Universal Data Access Web Site
Product Information
Technical Materials
Documentation
Downloads
Partners
Product Availability
Universal Data Access
OLE DB
ADO
ODBC
Remote Data Service (RDS) is a feature of ADO. RDS delivers a new Web data access technology that allows developers to create data-centric applications within ActiveX®-enabled browsers such as Microsoft® Internet Explorer.
ADO Visual C++ and Visual J++ Code Examples
These code examples demonstrate the usage of most ADO methods and properties with Microsoft Visual C++ and Microsoft Visual J++. These examples have been tested with MDAC 2.1.2.4202.3 (GA) and Visual Studio Service Pack 3. Versions of these code examples will be included in a future version of the ADO documentation.
Download:
Visual C++

Figura 1.- Información sobre *ADO* en *www.microsoft.com/data/ado*.

nueva versión, se ofrecen dos alternativas: el acceso a datos mediante *ADO* y los componentes *IB*.

ADO

Los componentes *ADO* encapsulan la funcionalidad del *framework ADO*. *ADO* (*Microsoft ActiveX Data Objects*) no es más que un conjunto de objetos que proporcionan a las aplicaciones la capacidad de acceder a diferentes fuentes de datos a través de *OLE DB*.

En palabras de *Microsoft*: "*ADO* proporciona una interfaz consistente y de alto rendimiento para el acceso a datos desde aplicaciones cliente, objetos de negocio en aplicaciones multicapa o incluso a navegadores *Web*. *ADO* es la interfaz que usted necesita conocer para el desarrollo de aplicaciones de *1-n* capas y basadas en *Web*".

OLE DB es la evolución natural de *ODBC*. Actualmente, *ODBC* es un estándar para el acceso a los datos contenidos en servidores de bases de datos relacionales. Con *OLE DB* se pretende dar un paso más, de tal forma que usando esta nueva tecnología, el acceso a cualquier fuente de datos sea genérico, tanto para una base de datos relacional como para un fichero de texto plano.

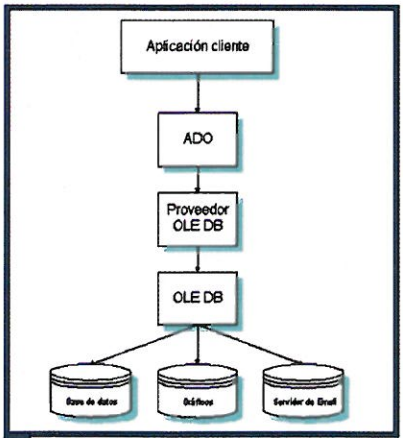


Figura 2.- Arquitectura ADO/OLE DB).

Citando de nuevo a *Microsoft*: "*ADO* se ha diseñado como una interfaz de alto nivel que permite acceder de una forma sencilla a los servicios proporcionados por *OLE DB*. Esta tecnología proporciona acceso de alto rendimiento a cualquier fuente de datos, incluyendo bases de datos relacionales, no relacionales, sistemas de ficheros, *email*, textos, gráficos, etcétera."

ARQUITECTURA

En la Figura 2 se observa la arquitectura de acceso a datos basada en *ADO/OLE DB*. Como se puede apreciar, existen los llamados *proveedores OLE DB*. En pasadas entregas vimos que *ODBC* es un estándar de acceso a datos, pero que no sirve de nada si no disponemos de controladores *ODBC* específicos para el gestor de bases de datos al que tengamos que acceder. Es decir, para acceder a *Oracle* mediante *ODBC*, necesitamos un controlador *ODBC* proporcionado por la misma *Oracle*. Pues con *OLE DB* ocurre lo mismo, ya que necesitamos controladores (proveedores) de cada una de las fuentes de datos a las que queramos acceder.

OLE DB permite acceder a cualquier fuente de datos

De todas formas, aún es posible usar *ADO* sin tener ningún proveedor *OLE DB*. Para ello, podemos recurrir a los controladores *ODBC*, puesto que existe un proveedor *OLE DB* genérico para las bases de datos *ODBC* relacionales. Es decir, gracias a este proveedor genérico podremos establecer un puente entre *ODBC* y *OLE DB*.

VERSIONES DE ADO

Desde su creación, han aparecido varias versiones de los objetos *ADO*. Actualmente, se encuentran por la versión *2.1*. Los componentes *ADO* de *Delphi* encapsulan a los objetos *ADO* versión *2.1*, por lo que éstos tendrán que estar instalados en la máquina donde se desarrolle, así como en las máquinas de los usuarios finales.

ADO 2.1 es la versión que necesitaremos para nuestras pruebas

Por cierto, resulta extremadamente importante que instaléis los parches que hay en la *Web* de *Borland* (*www.inprise.com*) para *Delphi 5*. Actualizan los componentes *ADO*, y al parecer, solucionan bastantes problemas que han vuelto locos a muchos usuarios.

PRINCIPALES COMPONENTES

Los principales objetos dentro de *ADO* son los objetos *conexión*, *comando* y *recordset*. El componente *TADOConnection* encapsula al objeto de conexión. Este componente se usa para establecer la conexión con la fuente de datos. Otros componentes *ADO* (como *TADOTable* o *TADOCommand*) pueden compartir la misma conexión. El equivalente a este componente en el *BDE* es *TDatabase*.

El componente *TADOCommand* encapsula al objeto comando. Permite ejecutar sentencias *SQL* que no devuelven conjuntos de datos (por ejemplo, un *insert* o un *delete*).

Estas sentencias *SQL* se pueden lanzar usando una conexión existente (creada mediante un *TADOConnection*), o creando una nueva (asignando los parámetros apropiados al componente *TADOCommand*).

Los principales objetos ADO: Conexión, Comando y Recordsets

Por último, para el acceso a los diferentes conjuntos de datos (*recordsets*), existen varios componentes. En primer lugar, tenemos *TADODataSet*, que puede considerarse como el principal componente usado para obtener datos y trabajar con ellos. No tiene equivalente en el *BDE*. Puede obtener datos de una o varias tablas simultáneamente.

El componente *TADOTable* es equivalente al componente *TTable* del *BDE*. Permite trabajar con datos almacenados en una tabla. El componente *TADOQuery* es equivalente a *TQuery*. Permite ejecutar cualquier sentencia *SQL*, tanto de consulta, inserción y modificación, como sentencias de creación de tablas, procedimientos almacenados, etcétera.

Por ultimo, *TADOStoredProc* (equivalente a *TStoredProc*) permite ejecutar procedimientos almacenados. Hay que indicar que todos los componentes pueden usar una conexión ya definida para acceder a los datos, o bien usar una nueva, que ellos mismos crean.

TRABAJAR CON ADO

Para ver cómo funcionan los componentes *ADO*, vamos a desarrollar una aplicación muy simple. Básicamente, rellena dos rejillas (*DBGrids*) a partir de una tabla y una consulta. Su aspecto puede verse en la Figura 4. Explicaremos el funcionamiento de cada componente siguiendo el orden en el que se insertan en la aplicación.

Figura 3.- *TADOConnection, TADOCommand, TADODataSet, TADOTable, TADOQuery y TADOStoreProc.*

TADOCONNECTION

Como hemos comentado, es el equivalente al *TDatabase*. Este componente permite conectar con los datos, de forma que la conexión creada pueda ser usada por el resto de los componentes *ADO*.

Los componentes ADO han sido diseñados para parecerse a los del BDE

Para crear una conexión con una fuente de datos, tendremos que rellenar la propiedad *Connection String*. El editor de propiedades asociado a esta propiedad permite construir la cadena de forma visual. Este editor no es propio de *Delphi*, sino que lo proporciona la propia interfaz *ADO*.

En nuestro caso, vamos a conectar con una base de datos *Access 97*, incluida en el *CD-ROM*. El acceso será mediante *ODBC –OLE DB*. Existe un proveedor *OLE DB* nativo para *Access*, pero a través de *ODBC* también podemos acceder a los datos (incluso si se encuentran en un formato no reconocido por *OLE DB*). Por supuesto, tendremos que crear previamente el *DSN* correspondiente (como vimos en entregas anteriores).

Para ello elegiremos la opción *Use Connection String -> Build*. En el cuadro de diálogo seleccionaremos las siguientes opciones: *Microsoft OLE DB Provider for ODBC Drivers* (pestaña *Provider*), *PRUEBAS_SOLOP* (el nombre de nuestro *DSN*, pestaña *Connection*), y en caso de que sea necesario, *login* y *password* de la base de datos.

La cadena de conexión generada deberá ser parecida a la siguiente:

```
Provider=MSDASQL.1;Persist
    Security Info=False;Data
    Source=PRUEBAS_SOLOP.
```

También es posible construir la cadena de conexión a mano (en tiempo de diseño o en tiempo de ejecución). Para ello hay que conocer los parámetros que la forman, recurriendo, si es necesario, a la ayuda de *Delphi*.

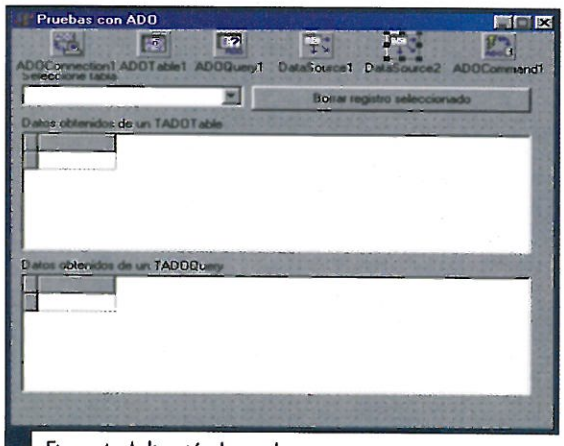


Figura 4.- Aplicación de prueba.

Mediante la propiedad *Connected*, podemos activar o desactivar la conexión. En caso de que no esté activa, tan pronto como alguno de los componentes que hagan uso de ella quiera acceder a los datos, la conexión se establecerá automáticamente. Los métodos *Open* y *Close* también permiten abrir o cerrar una conexión. El método *Open*, además, recibe como parámetros el *login/password* del usuario, aunque debemos indicar que no son obligatorios, puesto que éstos pueden ir en la cadena de conexión.

Los componentes pueden compartir una conexión o usar la suya propia

Hasta este punto se han explicado los conceptos básicos: cómo crear la cadena de conexión y cómo activarla. Veamos ahora algunas propiedades y métodos que pueden ser de gran utilidad. Por ejemplo, existen dos propiedades, *ConnectionTimeOut* y *CommandTimeOut* que permiten controlar cuánto tiempo pasará hasta que el sistema considere que ha habido un fallo al conectar con la base de datos, y ejecutar un comando, respectivamente.

Si los *timeouts* expiran, saltarán excepciones que permiten devolver, en caso de que sea necesario, el sistema a un estado seguro. Esto puede ser útil, por ejemplo, para cancelar las operaciones en caso de sobrecarga de los servidores o caídas de la red.

Para controlar las transacciones, contamos con los métodos *BeginTrans*, *CommitTrans* y *RollBackTrans*, equivalentes a las de *TDatabase*, por lo que remito al lector a las entregas anteriores para todo lo relacionado con este asunto.

TADOConnection proporciona el control de transacciones

Además, dispone de algunas propiedades útiles, como pueden ser *Datasets* y *DataSetCount*, que permiten conocer los *TADODatasets* relacionados por esta conexión. Existen propiedades parecidas para obtener los *TADOStoredProcs* y *TADOCommand's*

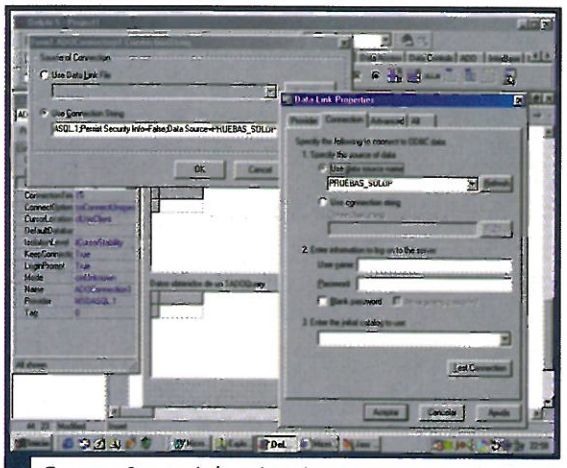


Figura 5.- Generando la cadena de conexión.

TADODATASET

Este componente permite acceder tanto a tablas, como a consultas, vistas, procedimientos almacenados, etcétera. Mediante las propiedades *CommandType* y *CommanText*, elegimos la fuente de los datos. Por ejemplo, si *CommandType= cmdTable* y *CommandText='empleados'*, estamos indicando que los datos provienen de la tabla *empleados*. Si *CommandType = cmdStoredProc* y *CommandText = 'Borrar_todos'* estamos indicando que queremos ejecutar el procedimiento almacenado *Borrar_todos*.

Es importante aplicar el parche para ADO

En nuestras pruebas no usaremos este componente. En su lugar, usaremos *TADOTable*, *TADOQuery* y *TADOStoredProc*. Estos componentes no son más que descendientes de *TADODataset*, sólo que en este caso, las propiedades *CommandType* vienen fijadas de antemano, y la propiedad *CommandText* se sustituye por *TableName*, *SQL*, o *ProcedureName*, haciendo que la similitud con sus equivalentes *VCL* sea total.

Con TADODataset podríamos reemplazar a TADOTable, TADOQuery y TADOStoredProc

Sea cual sea el componente que usemos, en todos los casos habrá que indicar qué conexión usar (propiedad *Connection*). Aunque es posible indicar que se use una nueva conexión (mediante la propiedad *ConnectionString* de cada componente), esto no es aconsejable para optimizar los recursos y el mantenimiento de la aplicación. Así que nosotros asignaremos a la propiedad *Connection* nuestro *TADO Connection*.

Como hemos dicho, insertaremos en el formulario un *TADO Table* y un *TADOQuery* (con sus correspondientes *TDataSource's*). Conectaremos los *DBGrids* a los *TDataSource*.

Aprovecharemos el método *OnChange* del *combo* para indicar qué tabla abrimos y qué consulta *SQL* lanzamos (como se puede ver en la Figura 7).

Podemos salvar los datos obtenidos en formato XML

Con respecto a la navegación por los datos, todo se realiza exactamente de la misma forma que con los componentes *BDE* (ya que éstos también descienden de la clase *TDataSet*). Es decir, usaremos los métodos *First, Last, Insert, Post, Append, Edit, etc.*, tal y como vimos en artículos anteriores. Con respecto a los eventos (*BeforePrint, BeforeDelete*, etc.), diremos que son idénticos a los de sus equivalentes *BDE*. También podremos crear relaciones *maestro/detalle* con la misma facilidad que con los componentes *BDE*.

Como curiosidad, señalaremos que los datos a los que hemos accedido con estos componentes pueden ser salvados o recuperados de un fichero en disco (que puede estar en nuestro ordenador o en cualquier otro de la red). Para ello, utilizaremos los métodos *Save ToFile* y *LoadFromFile*. El formato en el que se graban los datos puede ser *ADTG* y *XML* (esta segunda opción es, sin duda, la más interesante para el desarrollo de aplicaciones basadas en *Web*).

Por ultimo añadiremos un componente *TADOCommand*. Como dijimos, permite lanzar consultas *SQL* que no devuelvan resultados (como los *inserts*). En nuestro caso lo usaremos para vaciar la tabla seleccionada.

Este componente permite ejecutar consultas *SQL* y lanzar procedimientos almacenados. En la propiedad *CommandType* indicamos lo que queremos hacer (*cmdText, cmdStoredProc*), y en la propiedad *CommandText,* indicaremos la consulta *SQL* o el nombre del procedimiento almacenado. En nuestro caso, y puesto que queremos borrar la tabla seleccionada, tendremos que efectuar lo siguiente:

```
CommandType := cmdText;
CommanText:='Delete from TABLA'
```

Con ADO trabajaremos igual que con los componentes BDE

Después, basta con llamar al método *Execute*. Es posible llamar al método *Cancel* en caso de que la ejecución del comando quiera ser cancelada (siempre que éste aún no se haya ejecutado, claro está).

Con este componente, también podemos obtener conjuntos de datos, sin más que asignar el resultado devuelto por su método *Execute* a la propiedad *RecordSet* de cualquier *TADODataSet* o similar. Aunque la verdad, para obtener conjuntos de datos, ya tenemos otros componentes, por lo que éste sólo debería utilizarse para ejecutar ciertas sentencias *SQL*, aunque no dudo que en algunos casos esta característica podría resultar útil.

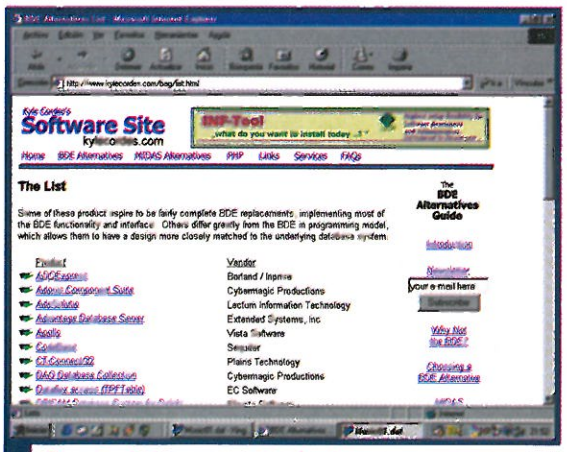


Figura 5.- Código que accede a los datos.

EJECUCIÓN DE LA PRUEBA

Sólo nos queda compilar y ejecutar. Si todo ha ido bien, habremos creado nuestra primera aplicación usando *ADO*, sin que haya excesivas diferencias con respecto al uso del *BDE*. Básicamente, una vez creada la cadena de conexión, se trabaja de la misma forma que con los componentes *BDE*. En el *CD-ROM* se encuentra el código de esta sencilla aplicación, así como la base de datos utilizada para su desarrollo

COMPONENTES IB EXPRESS

La otra alternativa al uso del *BDE,* la proporcionan los componentes *IB Express* (en adelante *IBX*). Con *ADO* podíamos acceder a cualquier fuente de datos, pero con *IBX* sólo podemos prescindir del *BDE* si nos limitamos a trabajar con *Interbase*.

Los componentes IBX se basan en los FreeIBComponents

Su gran ventaja es que ganamos velocidad. Seguimos dependiendo de *DLLs* externas, ya que en este caso, para que las aplicaciones fun-

cionen, debe encontrarse instalado en la máquina del usuario el cliente de *Interbase*.

En IBX se separa el control de transacciones de la gestión de conexiones

La *suite IBX* está basada en parte en un paquete de componentes de dominio público, muy conocidos en la comunidad desarrolladora: los *FreeIBComponets* (se pueden bajar de *www.interbase.com*). Puesto que son muy parecidos a los *IBX*, los lectores que hayan trabajado con ellos, no tendrán problemas al manejarlos.

En el resto del articulo iremos comentando uno a uno los diferentes componentes que integran la *suite IBX*. La mayoría son casi idénticos a sus equivalentes *BDE*, por lo que no nos pararemos mucho en ellos.

TIBDATABASE

Este componente es el equivalente al *TDatabase*. Se utiliza para establecer la conexión con la base de datos. A diferencia de la versión *BDE*, este componente no gestiona las transacciones. Para ello existe un componente específico: *TIB Transaction*.

En la propiedad *DataBaseName* introduciremos la base de datos a la que nos queremos conectar, con el siguiente formato:

```
Base de datos local:
    Unidad:\directorio\nombre_fi
    chero.gdb
Base de datos remota (TCP/IP):
    servidor:nombre_fichero.gdb
Base de datos remota (NetBEUI):
    \\servidor\fichero.gdb
```

Por lo demás, el funcionamiento es el mismo que su equivalente *BDE*, por lo que remito al lector a anteriores entregas.

TIBTRANSACTION

Este componente permite gestionar las transacciones. La ventaja de separar la gestión de transacciones de la gestión de conexiones, es que con una única conexión podemos gestionar múltiples transacciones (con el consiguiente ahorro de recursos).

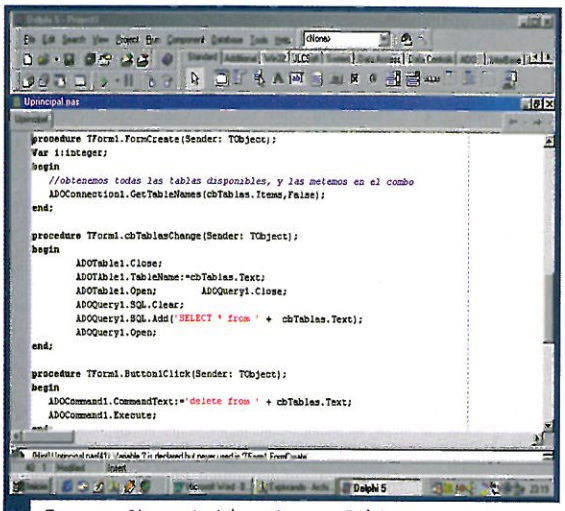


Figura 6.- Obtención del parche para *Delphi 5*.

El caso más normal, es decir, una conexión, o una transacción, se implementa asignando a la propiedad *DefaultTransacction* del *TIBDatabase* un componente *TIBTransacction*.

Si queremos asignar más transacciones a una misma conexión, tendremos que insertar nuevos componentes *TIBTransaction*, y añadir estas transacciones a la conexión, mediante el método *TIBDatabase.AddTransaction(Transaccion: TIBTransaction)*. De igual forma, también podemos eliminar transacciones de la conexión (usando para ello el método *RemoveTransacction*, al que se le pasa como parámetro el índice de la transacción). En cualquier momento, mediante las propiedades del componente *TIBDatabase Transactions* y *TransactionsCount* podemos conocer las transacciones que van a través de una conexión.

También podemos realizar el proceso inverso. Es decir, podemos indicar a un *TIBTransaction* qué bases de datos van a ser gestionadas por este componente. Para ello, contamos con los métodos *Add Database*, *RemoveDatabase* y *Find DataBase*. En todo momento, gracias a las propiedades *Databases* y *DatabaseCount*, podemos saber qué bases de datos usan esta transacción. Podemos asignar una base de datos por defecto mediante la propiedad *DefaultDatabase*.

Es posible manejar varias transacciones con una misma conexión

Y por supuesto, todos los métodos de *TDatabase* relacionados con las transacciones (*Commit*, *Roll back*, etc.), ahora pasan a ser métodos de *TIBTransacction*.

EL PRÓXIMO MES

Hasta aquí, hemos visto los aspectos más importantes de *Ado* e *IB* Express. En el próximo número, seguiremos profundizando en el uso de los componentes *IBX*. Además, veremos como utilizar un mecanismo muy potente, las actualizaciones en caché.

Configuración de X-Window en Linux

Antonio José Novillo López.

Con el paso del tiempo, una de las mejoras sustanciales que han ido apareciendo en el mundo del *Linux,* han sido las herramientas gráficas de configuración. En este artículo veremos cómo operan dichas herramientas en la configuración del entorno *X-Window,* así como las variaciones encontradas durante el proceso de instalación de la nueva versión 4.0 de *XFree86.*

INTRODUCCIÓN

El tiempo en el que había que editar los ficheros de configuración desde la línea de comandos ha pasado ya, ¡gracias a Dios! Para los que lo hemos vivido, configurar el entorno gráfico de *Linux* era poco menos que un tormento... Un fichero de texto lleno en multitud de ocasiones de opciones crípticas, un editor, y a probar.

¡Qué lejos quedan aquellos días, en los que disponíamos de potentes herramientas capaces de detectar por sí solas el *hardware* del sistema y realizar una configuración automática! Este avance redundará, sin duda, en el acercamiento del sistema a los usuarios menos avanzados, evitándoles las tareas más complejas. Hoy por hoy es bastante raro tener que editar el fichero de configuración de *XFree* a mano, ya que sólo es necesario en casos muy particulares y normalmente debido a *"bugs"* en alguna aplicación de configuración.

El sistema de ventanas *XFree 3.x* está formado por una fichero ejecutable denominado *servidor*, que es el que recibe las conexiones de los clientes y una serie de librerías. A diferencia de otros entornos gráficos que permiten la carga dinámica de *drivers* para las tarjetas gráficas, esta versión del sistema de ventanas viene con un servidor diferente para cada tarjeta gráfica.

Para trabajar de momento es conveniente seguir utilizando las versiones anteriores

Por ejemplo, los usuarios de tarjetas con el *chipset Mach32* tendrán que utilizar el servidor específico para esta tarjeta. Se pueden tener instalados en el sistema varios servidores, de tal forma que se ejecutará aquel que esté enlazado simbólicamente con el fichero *X*. De esta forma, si enlazamos *X* con el servi-

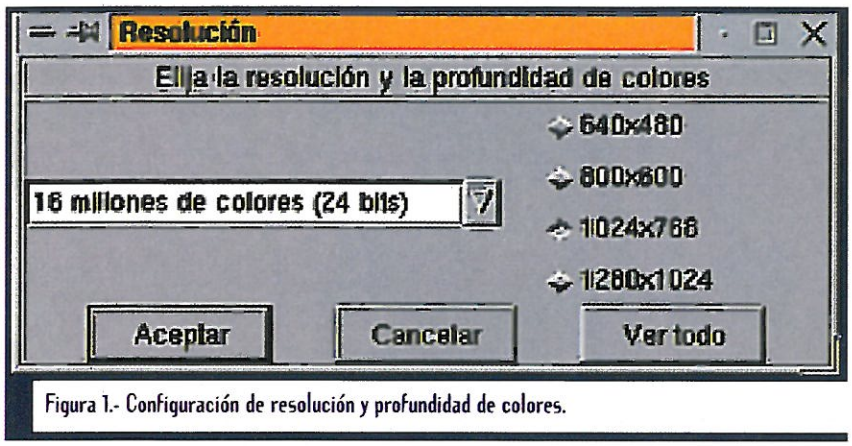


Figura 1.- Configuración de resolución y profundidad de colores.

dor para *SVGA* genérica, este será el servidor que se ejecute.

EL FICHERO DE CONFIGURACIÓN DE XFREE 3.X

Toda la información relativa a la configuración de un servidor gráfico se puede encontrar en el fichero */etc/XF86Config*. La información incluida en este fichero es, en su mayoría, independiente del servidor gráfico que estemos utilizando, por lo que si cambiamos de tarjeta gráfica, el sistema no se verá afectado gravemente. Tan sólo pueden alterarse los distintos *relojes* de los modos gráficos.

El fichero está compuesto por diversas secciones, cada una de las cuales refleja el funcionamiento de uno de los dispositivos del sistema. Así, podremos disponer de secciones para el monitor, la tarjeta gráfica, el teclado, el ratón y demás dispositivos.

En la Figura 2 podemos encontrar una parte de un fichero típico de configuración, en el que se aprecian las distintas secciones. Como se puede ver, cada sección puede disponer a su vez de subsecciones que irán definiendo el comportamiento del dispositivo. En el caso de una tarjeta gráfica, la sección será *Device*, donde se especificarán parámetros como los *relojes* o la memoria de la tarjeta.

Existe otra sección llamada *Screen* donde terminaremos de definir el comportamiento gráfico de nuestro servidor. En esta sección indicaremos qué tarjeta gráfica, de las anteriormente definidas, utilizamos, así como los modos gráficos que deseamos usar. También indicaremos cuál es el monitor con el que trabajamos. En la sección de monitor se añaden los valores de sincronismo horizontal y vertical, así como los modos de trabajo. Posteriormente, en la sección *Screen* se pueden escoger estos modos para cada profundidad de color.

Sin lugar a dudas, este método de configuración es notablemente complejo, e implica tener que incluir valores que no siempre conocemos. Por esto, el sistema incluye desde hace bastante tiempo una herramienta de modo texto para esta configuración, *XF86Config*. Mediante una serie de preguntas, el programa irá presentando las distintas posibilidades de configuración de cada dispositivo: modelo de tarjeta gráfica, monitor, frecuencias, resoluciones, etcétera. No se puede decir que sea un programa terriblemente cómodo, pero facilita el trabajo. Su uso es muy recomendable cuando no dispongamos de ninguna herramienta gráfica capaz de configurar el sistema.

HERRAMIENTAS DE CONFIGURACIÓN GRÁFICA

La herramienta gráfica de configuración de *XFree86* por defecto es *XF86Setup*. Este programa presentará una serie de ventanas de diálogo en modo gráfico, que harán que la configuración del sistema sea cosa de niños. Para lanzarla, tan sólo tendremos que ejecutar desde la línea de comandos:

```
> XFree86Setup
***F/C
```

El sistema efectúa las tareas de configuración más complicadas, sin la intervención del usuario

A continuación aparecerá una ventana gráfica a modo de asistente donde se nos irá preguntando por la configuración de nuestro sistema. Iremos configurando el ratón, el teclado, la tarjeta gráfica, el monitor y los parámetros de profundidad de color y resolución. Tras seleccionar estos valores, el programa lanzará una prueba para ver si el funcionamiento es correcto. De ser así, grabará un fichero *XF86Config*.

Posteriormente, podremos utilizar la herramienta *xvidtune* para ajustar con mayor detalle la confi-

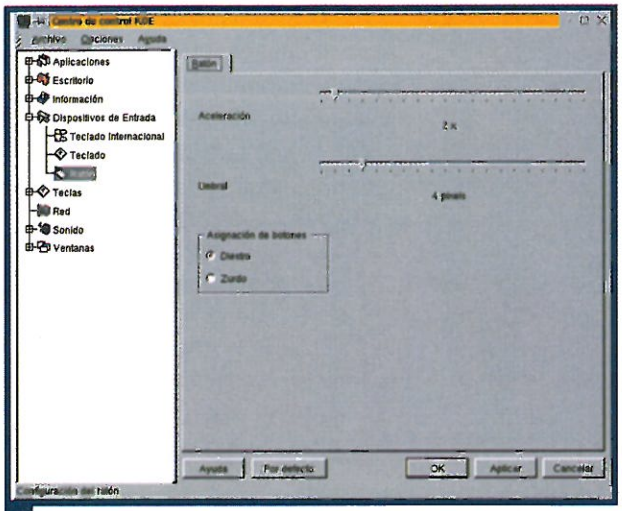


Figura 2.- Configuración del ratón desde el *Centro de control*.

guración del monitor. De esta forma podremos ajustar el ancho y alto de la pantalla, así como centrar la imagen en el monitor.

En las últimas versiones de instaladores que incluyen las distribuciones, están surgiendo herramientas capaces de detectar el *hardware* del sistema de forma automática y configurarlo. Aunque generalmente se nos preguntará por la profundidad de colores y la resolución del sistema, estos valores los podremos modificar posteriormente, con lo que no tendremos que lanzar ninguna herramienta, y el sistema estará configurado adecuadamente tras su instalación.

Estos sistemas automáticos son un gran avance, aunque si no son capaces de detectar una tarjeta correctamente, pueden instalar un servidor erróneo que dé problemas cuando vaya a ejecutarse. Es por tanto muy conveniente conocer el procedimiento de configuración manual de la tarjeta, por si surgen estos problemas. Este caso se da con bastante frecuencia si queremos instalar *Linux* en máquinas virtuales del tipo *VMWare*.

Por otra parte, nos encontramos con una de las grandes carencias de *XFree 3.x*. Al requerir un servidor distinto para cada tarjeta gráfica, si tenemos una tarjeta más moderna y no soportada, tendremos problemas para lanzar el sistema gráfico. Por un lado, deberíamos encontrar un servidor que soportara una nueva tarjeta o esperar a que salga una versión nueva de *XFree* que la soporte.

XVIDTUNE servirá para hacer los últimos ajustes de monitor

Otra opción consiste en utilizar las capacidades de *framebuffer* que aparecieron con la versión *2.2* del *kernel*, que permiten utilizar un servidor genérico para todas aquellas tarjetas que soporten *VESA 2.0*. Sin embargo, la instalación se complica pues tendremos que recompilar el núcleo y realizar modificaciones adicionales a las anteriormente mencionadas. De todas formas, como veremos más adelante, esta dificultad se ha superado con la versión *4* de *XFree*.

MODIFICACIÓN DE PARÁMETROS EN CALIENTE

Una vez que comenzamos a trabajar con nuestro entorno gráfico, puede surgirnos la necesidad de cambiar algunas de las propiedades configuradas con anterioridad en el sistema. Habitualmente, cambiaremos alguna propiedad del ratón (velocidad del puntero en pantalla), la profundidad de color o la resolución. Existen métodos que permiten modificar estos valores sin necesidad de variar el fichero de configuración o reiniciar el entorno gráfico.

La aplicación XF86Setup que acompaña a XFree permite configurar gráficamente el sistema

Está claro que el método más sencillo de cambiar parámetros consiste en salir del entorno gráfico (en el peor de los casos tendremos que "matar" el programa *X*), modificar la configuración y volver a lanzar el programa. El servidor *X* coge la configuración cada vez que arranca. Casi todas las nuevas distribuciones incluyen dentro de sus herramientas de configuración programas capaces de modificar estos valores de una forma cómoda e intuitiva. En general, estos programas serán capaces de modificar la configuración, pero exigen un reinicio del entorno gráfico para llevar a cabo los cambios.

Por ejemplo, *Mandrake7* incluye *DrakX*, que permite realizar este proceso de configuración. Casi todas las distribuciones incluyen su propia herramienta, lo que implica que deberemos conocer el funcionamiento de la distribución con la que estemos trabajando.

Por otro lado, el panel de control de *KDE* incluye algunas herramientas capaces de modificar el comportamiento de algunos dispositivos. Desgraciadamente, las

mentación de *OpenGL* y *GLX* que permite utilizar aceleradoras *3D* directamente. El soporte todavía es parcial y aunque están anunciadas varias tarjetas, aún han de ser implementadas varias más.

La configuración de XFree se almacena en el fichero XF86Config

Asimismo, se ha integrado un servidor de fuentes capaz de procesar fuentes del tipo *truetype*, con lo que podremos añadir dinámicamente nuevas fuentes (existen innumerables gratuitas en *Internet*) a nuestro entorno.

Otra novedad bastante importante ha sido la inclusión de soporte para *Xinerama*. Este nombre tan extraño tiene como significado la posibilidad de mostrar el contenido de nuestro servidor en varios monitores, como si de una pantalla de cine se tratara.

Existen muchas más mejoras, incluyendo rendimientos y mejoras transparentes al usuario, pero las anteriormente comentadas representan las más destacables.

MODIFICACIONES EN INSTALACIÓN DE XFREE 4.0

Otro de los cambios más influyentes que se han realizado en esta versión, ha sido el nuevo formato del fichero *XF86Config*. Esto conlleva que los programas de configuración anteriores dejan de ser compatibles. Asimismo, la nueva distribución no incluye un programa gráfico como el anterior *XF86Setup*, lo cual complica un tanto la configuración. Es de suponer que cuando las grandes distribuciones adopten la versión 4 como definitiva, sus programas de configuración serán capaces de facilitarnos el trabajo.

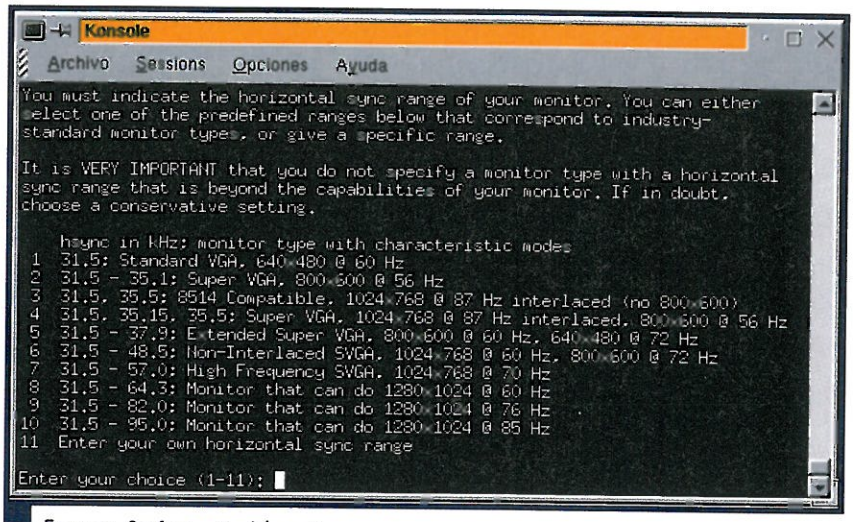

Konsole
Archivo Sessions Opciones Ayuda
You must indicate the horizontal sync range of your monitor. You can either
select one of the predefined ranges below that correspond to industry-
standard monitor types, or give a specific range.
It is VERY IMPORTANT that you do not specify a monitor type with a horizontal
sync range that is beyond the capabilities of your monitor. If in doubt,
choose a conservative setting.
hsync in kHz; monitor type with characteristic modes
1 31.5; Standard VGA, 640x480 @ 60 Hz
2 31.5 - 35.1; Super VGA, 800x600 @ 56 Hz
3 31.5, 35.5; 8514 Compatible, 1024x768 @ 87 Hz interlaced (no 800x600)
4 31.5, 35.15, 35.5; Super VGA, 1024x768 @ 87 Hz interlaced, 800x600 @ 56 Hz
5 31.5 - 37.9; Extended Super VGA, 800x600 @ 60 Hz, 640x480 @ 72 Hz
6 31.5 - 48.5; Non-Interlaced SVGA, 1024x768 @ 60 Hz, 800x600 @ 72 Hz
7 31.5 - 57.0; High Frequency SVGA, 1024x768 @ 70 Hz
8 31.5 - 64.3; Monitor that can do 1280x1024 @ 60 Hz
9 31.5 - 82.0; Monitor that can do 1280x1024 @ 76 Hz
10 31.5 - 95.0; Monitor that can do 1280x1024 @ 85 Hz
11 Enter your own horizontal sync range
Enter your choice (1-11):

Figura 5.- Configuración del monitor.

Hasta entonces, tendremos que volver a los métodos de consola. De todas formas, no es recomendable utilizar esta nueva versión para el día a día, sino más como curiosidad de cara a descubrir sus nuevas propiedades.

Existe una herramienta de configuración, la anteriormente mencionado *XF86Config*, que no deja de ser una adaptación de la versión anterior con mejoras para soportar el nuevo formato de fichero de configuración. El programa tiene fallos, pero facilita en cierto modo el proceso.

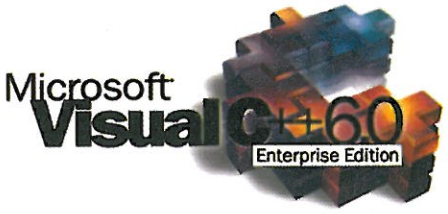
Por otro lado, se puede intentar realizar una configuración automática con el comando *XFree86 –configure*, que genera el fichero *$HOME/XF86Config.new*. Desgraciadamente, si falla la detección de alguno de los dispositivos, el programa es incapaz de realizar la configuración. Por eso, la solución más completa consiste en editar un fichero de configuración a partir del fichero de ejemplo en */usr/X11R6/lib/XF86Config.eg*. Las reglas son similares a las de la versión antigua, estando cada sección documentada.

CONCLUSIÓN

A lo largo del artículo, hemos visto las bases de la configuración del entorno gráfico así como los métodos necesarios para ello. Como en casi todos los entornos, es fundamental conocer la base teórica que hay detrás, el fichero de texto plano. Una vez conocida, podemos lanzarnos al uso de herramientas gráficas. De esta forma, podremos configurar cómodamente, pero cuando surjan problemas seremos capaces de reaccionar y resolverlos.

Asimismo, hemos descubierto las herramientas de que disponemos a la hora de modificar los parámetros de funcionamiento.

Por último, se han adelantado las novedades la nueva versión *4.0*, teniendo siempre en mente que su uso por el público en general será inmediato.



Visual C++ y MFC (y VI)

Constantino Sánchez Ballesteros.
Técnico Superior de Desarrollo de Aplicaciones Informáticas.

Terminamos la serie de artículos sobre *MFC's* centrándonos en la creación de controles propios. En particular, vamos a referirnos a los botones sensibles al ratón, en sus diferentes estilos, funciones y aspecto.

CONTROLES PERSONALIZADOS

Los controles personalizados, también conocidos como *Owner-Drawn* son aquellos que tienen la misma función al utilizarlos que la de los controles base de que dependen, pero que pueden contener implementaciones del programador para cambiar su aspecto, o alguna funcion especial añadida.

Nuestro objetivo en este artículo consistirá en la creación de un botón sensible, de manera que tenga tres estados:

- Pulsado.
- No pulsado.
- Seleccionado.

El estado seleccionado será algo vistoso, ya que cuando pongamos el ratón sobre nuestro botón, éste se activará de forma automática. Para ello, se han creado tres *bitmaps* que representarán los diferentes estados en los que se encuentra el botón (*Figura 1*).

Como ya se comentó, las aplicaciones pueden crear controles personalizados para realizar tareas no implementadas en las clases de los controles que incluye por defecto el entorno de desarrollo de *Visual C++*. La *API Win32* ofrece tres formas de crear controles personalizados:

1. Con el uso de controles *owner-drawn*.
2. Subclasificando la clase *Window* de un control existente.
3. Implementando una clase *Window* propia.

CONTROLES OWNER-DRAWN

Los botones, las listas y cuadros combinados tienen estilos *owner-drawn,* que permiten que el control envíe un mensaje a la ventana que los contiene (*Parent*) cada vez que el control tenga que ser dibujado. Esta característica permite alterar la apariencia de un control, personalizándolo en función de cada aplicación desarrollada.

Los estilos de un control definen su funcionalidad y aspecto visual

El estilo *owner-drawn* afecta a los botones, según la forma en que el sistema dibuja esos botones. Para las listas y cuadros combinados, el sistema dibuja los elementos contenidos por el control, y el control dibuja su propio marco. Por ejemplo, una aplicación puede crear un cuadro de lista propio para dibujar un *bitmap* al lado de cada elemento contenido en la lista.

Una aplicación puede diseñar cuadros de lista, combinados y botones como *owner-drawn* creándolos con el estilo apropiado.

ESTILOS DE LOS BOTONES

Cuando creamos botones, podemos especificar una combinación de los siguientes estilos para establecer su funcionalidad y aspecto:

- *BS_3STATE*: crea un botón igual a una casilla de verificación, excepto que la caja puede ser bloqueada en tono gris, como marcada o desactivada. El estado *Grayed* significa que el estado del botón es indeterminado.
- *BS_AUTO3STATE*: crea un botón igual a una casilla de verificación tri-estado, excepto que la caja cambia su estado cuando el usuario la selecciona. El estado forma un bucle a través de los estados marcado, *Grayed*, y no marcado.
- *BS_AUTOCHECKBOX*: crea un botón como una casilla de verificación, pero en la que el estado cambia automáticamente entre marcado y no marcado cada vez que el usuario la selecciona.
- *BS_AUTORADIOBUTTON*: crea un botón como el de radio, excepto que cuando se selecciona, el sistema establece automáticamente el estado del botón a marcado y los demás controles del grupo como no marcados.
- *BS_CHECKBOX*: crea una pequeña casilla de verificación con texto a su derecha. Para visualizar el texto a la izquierda de la casilla debemos combinar este *flag* con el estilo *BS_LEFTTEXT*.
- *BS_DEFPUSHBUTTON*: crea un botón de pulsación que tiene un borde marcado en color negro. Si el botón se encuentra en una caja de diálogo, podemos seleccionarlo presionando la tecla ENTER, aún cuando el botón no tenga el foco en ese momento. Este estilo es útil para hacer selecciones por defecto cómodamente.
- *BS_GROUPBOX*: crea un rectángulo que puede contener controles agrupados. Cualquier texto asociado con este estilo se ve en la esquina superior izquierda del rectángulo.
- *BS_LEFTTEXT*: coloca el texto en el lado izquierdo del botón de radio o casilla de verificación.
- *BS_OWNERDRAW*: este es el estilo que utilizaremos para poder dibujar el botón a nuestro gusto. La ventana que forma el control recibe un mensaje *WM_MEASUREITEM* cuando se crea el botón y un mensaje *WM_DRAWITEM* cuando cambia su aspecto visual. No se debe combinar el estilo *BS_OWNERDRAW* con ningún otro estilo de botón.
- *BS_PUSHBUTTON*: crea un botón de pulsación que envía un mensaje *WM_COMMAND* a la ventana del botón cuando el usuario lo selecciona.
- *BS_RADIOBUTTON*: crea un pequeño círculo con texto. Por defecto, el texto se visualizará a la derecha del control. Para visualizarlo a la izquierda debemos combinar con el estilo *BS_LEFTTEXT*.
- *BS_BITMAP*: especifica que el botón visualiza *bitmaps*.
- *BS_BOTTOM*: coloca el texto en la parte baja del rectángulo del botón.
- *BS_CENTER*: centra el texto horizontalmente en el rectángulo del botón.
- *BS_ICON:* se visualizará un icono en el botón.
- *BS_FLAT*: el botón se representa en dos dimensiones.
- *BS_LEFT*: justifica el texto del botón a la izquierda.

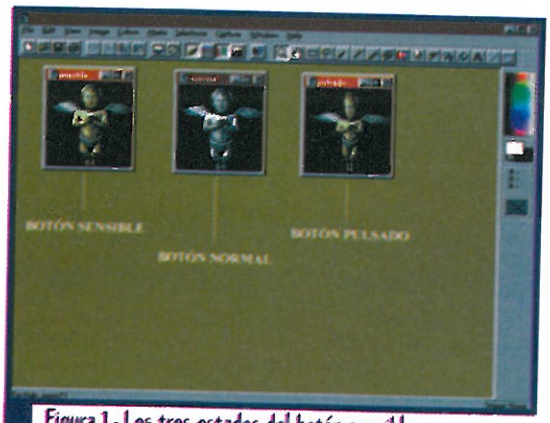


Figura 1.- Los tres estados del botón sensible.

- *BS_MULTILINE*: activa el texto del botón con soporte multilínea para que el texto que no quepa en el ancho del botón se visualice en otra línea.
- *BS_NOTIFY*: permite a un botón enviar notificaciones *BN_DBLCLK, BN_KILLFOCUS* y *BN_SETFOCUS* a su ventana pariente.
- *BS_PUSHLIKE*: permite que un botón (casilla de verificación, botón radio, etc.) actúe como un botón de pulsación.
- *BS_RIGHT*: justifica el texto a la derecha.
- *BS_TEXT*: especifica que el botón visualizará texto.
- *BS_TOP*: coloca texto en la parte superior del rectángulo del botón.
- *BS_VCENTER*: coloca el texto en medio del rectángulo que ocupa el botón.

Para crear un botón personalizado deberemos crear una clase derivada de cbutton

Para hacer nuestro botón, debemos recordar que se creará en tiempo de ejecución, y por lo tanto, no podremos insertarlo como un botón normal dentro del editor de recursos.

LISTADO 1. Propiedades y métodos.

```
Propiedades y métodos.*****
class CMiBoton : public CButton
{
// Construction
public:
  CMiBoton();

// Atributos
private:
  bool pulsado;
  bool parent;
  CDC *m_dc_pulsado,*m_dc_nopulsado,*m_dc_sensible;
  int xpos,ypos;
  CBitmap btdown,btup,btsense;

// métodos
public:
  void TestButton(CPoint point);
  void SelectBitmap(UINT ID_PULSADO,
      UINT ID_SENSIBLE,UINT ID_NOPULSADO);
```

Esto significa que tendremos que crearlo por código, estableciendo las propiedades oportunas. El botón será diseñado mediante el soporte de una clase creada por nosotros, derivada de la clase *CButton*.

CREACIÓN DEL PROYECTO

Para empezar el proyecto, debemos crear una aplicación desde el asistente con soporte *MFC's*. Una vez generado el esqueleto de la aplicación, tendremos que crear una nueva clase (*CMiBoton*) desde el asistente, que tendremos que derivarla de la clase *CButton*. En la Figura *2* podemos ver una captura de la ventana donde creamos nuestra clase personalizada.

Una vez creada la clase, añadiremos las implementaciones necesarias. En la cabecera de la clase debemos establecer las funciones y variables que utilizaremos, cuyas propiedades y métodos podemos apreciar en el Listado 1.

Como se ve, se crearán dos métodos públicos para trabajar con nuestro botón. *TestButton* se encargará de enviar la coordenada del ratón cuando se encuentre apuntando a la ventana que contiene nuestro botón. De esta manera podremos representar el botón sensible cuando el puntero del ratón se sitúe encima de nuestro botón.

Hay un problema bastante importante, y es que cuando obtenemos el puntero del ratón desde nuestra clase *CmiButton*, obtendremos la coordenada relativa del ratón a partir del rectángulo que ocupa el propio botón. Esto significa que si el botón lo hemos creado a partir de la posición *(50,50)* de la ventana, y si el puntero se sitúa dentro de esa coordenada, realmente la coordenada que devolverá será *(0,0)*. Para resolver esta situación, tendremos que pasar constantemente la posición del ratón desde dentro de nuestra ventana (*Vista*), de tal manera que podremos determinar si el ratón debe activarse automáticamente o no.

Los contextos de dispositivo podemos utilizarlos para realizar operaciones gráficas sobre ellos

Cuando creemos un objeto de nuestra clase, se llamará al constructor de la misma. Este es un buen lugar para crear en memoria los *DC's* correspondientes.

```
CMiBoton::CMiBoton()
{
    m_dc_pulsado=new CDC;
    m_dc_nopulsado=new CDC;
    m_dc_sensible=new CDC;
    pulsado=false;
}
```

Del mismo modo, cuando se destruya la clase (*destructor*), será con-

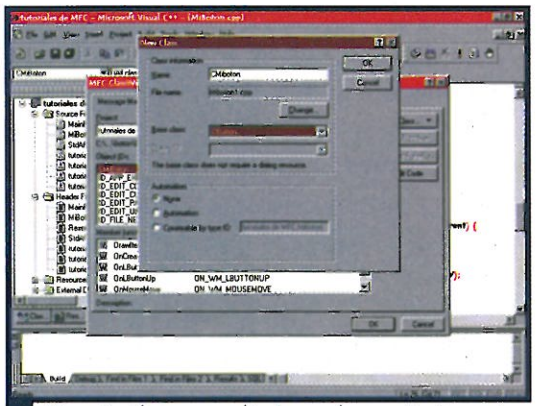


Figura 2.- Clase personalizada para la creación de nuestro botón sensible.

veniente liberar la memoria ocupada por dichos *DC's* para dejar al sistema libre de recursos utilizados:

```
CMiBoton::~CMiBoton()
{
    delete m_dc_pulsado;
    delete m_dc_nopulsado;
    delete m_dc_sensible;
}
```

Cuando el programador decida crear un botón dentro de la ventana, se llamará al evento *OnCreate*. En este evento crearemos los *DC's* para poder trabajar con ellos. Recordemos que anteriormente, en el constructor de la clase sólo les reservamos memoria.

```
int CMiBoton::OnCreate(LPCREATES-
    TRUCT lpCreateStruct)
{
   if
    (CButton::OnCreate(lpCreateSt
    ruct) == -1)
      return -1;
   m_dc_pulsado-
    >CreateCompatibleDC (GetDC
    ());
   m_dc_nopulsado-
    >CreateCompatibleDC
    (GetDC());
   m_dc_sensible-
    >CreateCompatibleDC
    (GetDC());

   return 0;
}
```

CreateCompatibleDC permitirá que los nuevos *DC's* sean compatibles con el del botón creado para que no haya problemas al dibujar en su *DC*.

Cada vez que se mueva el ratón desde dentro de los límites de nuestro botón debemos hacer una llamada a la función *DrawItem*. Esta función se encargará de dibujar el *bitmap* correspondiente, dependiendo del estado en que se encuentre el botón (activado, pulsado, no pulsado). Es necesario pasar a *false* la variable *parent*, para saber que la posición del ratón que hemos obtenido se refiere a la relativa al botón y no a la absoluta de la ventana.

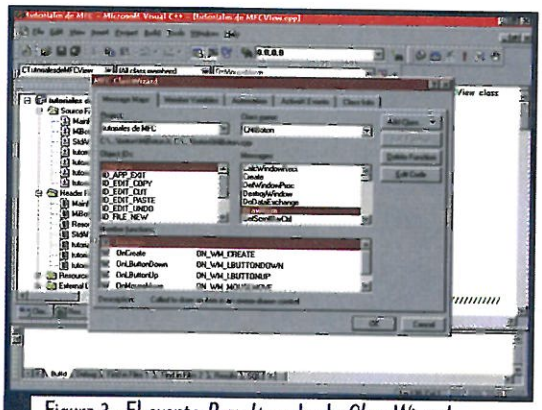


Figura 3.- El evento *DrawItem* desde *ClassWizard*.

```
void
    CMiBoton::OnMouseMove(UINT
    nFlags, CPoint point)
{
   xpos=point.x;ypos=point.y;
   parent=false;
   DrawItem(NULL);
   CButton::OnMouseMove(nFlags,
    point);
}
```

SelectBitmap se utiliza para pasarle a la clase los tres *bitmaps* que se representarán, dependiendo del estado de nuestro botón. Por supuesto, estos *bitmaps* podremos cargarlos desde el editor de recursos y pasar sus *ID's* a esta función.

```
void CMiBoton::SelectBitmap(UINT
    ID_PULSADO,
              UINT
    ID_SENSIBLE,UINT ID_NOPULSA-
    DO)
{
   SetButtonStyle( WS_VISIBLE |
    BS_OWNERDRAW, TRUE );
   if (ID_PULSADO) {
      btdown.LoadBitmap(ID_PULSA-
    DO);
      m_dc_pulsado->SelectObject
    (&btdown);
   }

   if (ID_NOPULSADO) {
      btup.LoadBitmap(ID_NOPULSA-
    DO);
      m_dc_nopulsado-
    >SelectObject (&btup);
   }
   if (ID_SENSIBLE) {
      btsense.LoadBitmap(ID_SEN-
    SIBLE);
      m_dc_sensible->SelectObject
    (&btsense);
   }
}
```

Como podemos ver, cuando llamemos a esta función, utilizaremos el método *SetButtonStyle* para establecer el estilo de nuestro botón como *BS_OWNERDRAW*. Así, podremos implementar nuestras propias rutinas de dibujo.

La variable *Pulsado* se utiliza para saber si el botón está pulsado o no, mientras que la variable *Parent* se utiliza para saber si la coordenada del ratón viene de la vista de nuestro programa o del del evento *OnMouseMove* del propio botón.

Las variables de tipo *CBitmap* se encargarán de cargar los *bitmaps* pasados a *SelectBitmap* y asociarlos con tres *DC's* (clase *CDC* *) para poder representar los gráficos posteriormente. Recordemos que un *DC* es un contexto de dispositivo que podemos utilizar para por ejemplo, dibujar en él. Un *DC* puede representar a la pantalla, impresora, etcétera. Aquí, lo utilizaremos como soporte gráfico de pantalla.

Utilizando los eventos que proporciona la clase *CButton* referentes al ratón sabremos si el botón izquierdo del ratón ha sido pulsado o no. De este modo podremos dar el valor adecuado a la variable *pulsado*.

```
void CMiBoton::OnLButtonDown(UINT
     nFlags, CPoint point)
{
   pulsado=true;
   CButton::OnLButtonDown(nFlags,
    point);
}

void CMiBoton::OnLButtonUp(UINT
     nFlags, CPoint point)
{
   pulsado=false;
   CButton::OnLButtonUp(nFlags,
    point);
}
```

El mensaje wm-drawitem indica cuándo podemos comenzar las operaciones de dibujo del control

La función más importante de la clase es, sin lugar a dudas, *DrawItem*, la que corresponde al evento *WM_DRAWITEM* del control, usada para dibujar en pantalla el *bitmap* que corresponda. Su funcionamiento es el siguiente:

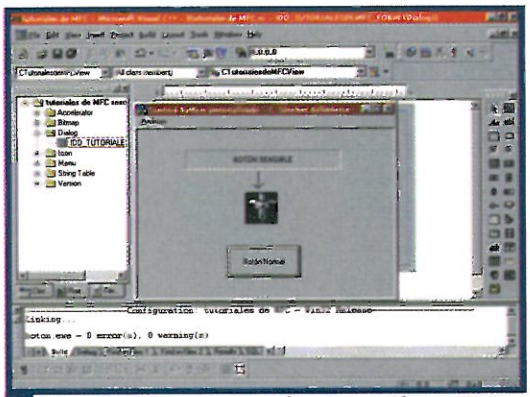


Figura 4.- Diferencias entre el botón sensible y un botón normal agregado desde el editor de recursos.

TABLA 1.

StretchBlt

```
BOOL StretchBlt(
  HDC hdcDest,      // DC destino
  int nXOriginDest, // coordenada X destino de la esquina superior izquierda
  int nYOriginDest, // coordenada Y destino de la esquina superior izquierda
  int nWidthDest,   // ancho del rectángulo destino
  int nHeightDest,  // alto del rectángulo destino
  HDC hdcSrc,       // DC origen
  int nXOriginSrc,  // coordenada X destino de la esquina superior izquierda
  int nYOriginSrc,  // coordenada Y destino de la esquina superior izquierda
  int nWidthSrc,    // ancho del rectángulo origen
  int nHeightSrc,   // alto del rectángulo destino
  DWORD dwRop       // operación raster utilizada
);
```

- Si la variable *pulsado* tiene el valor *true,* significa que el botón del ratón ha sido pulsado sobre el botón, y por lo tanto, tendremos que pintar el *bitmap* utilizado para ese estado.
- Si la variable *pulsado* tiene el valor *false* significa que el botón del ratón no ha sido pulsado sobre nuestro control, y también tendremos que ver el gráfico que corresponda.
- Finalmente, si *parent* tiene el valor *true* significa que el puntero del ratón no está dentro de los límites del rectángulo del botón, y por lo tanto será otro estado diferente para el botón. Del mismo modo, se dibujará el otro *bitmap* que queda.

```
void
CMiBoton::DrawItem(LPDRA-
WITEMSTRUCT
lpDrawItemStruct)
{
   CDC *dc=GetDC ();
   CRect rect;
    rect.top=rect.left=0;
    rect.bottom
=rect.right=30;
CBrush brush;
    GetClientRect (rect);
 //tamaño del control

if (pulsado) {
  dc-
 >StretchBlt(0,0,rect.right,re
 ct.bottom,m_dc_pulsado,

 0,0,65,65,SRCCOPY);
}
else {
   if (xpos<rect.right  &&
 xpos>rect.left  &&
 ypos<rect.bottom &&
       ypos>rect.top &&
 !parent) {
       dc-
 >StretchBlt(0,0,rect.right,re
 ct.bottom,m_dc_sensible,

 0,0,65,65,SRCCOPY);
   }
  else if (parent) {
     dc-
 >StretchBlt(0,0,rect.right,re
 ct.bottom,m_dc_nopulsado,

 0,0,65,65,SRCCOPY);
  }
}
}
```

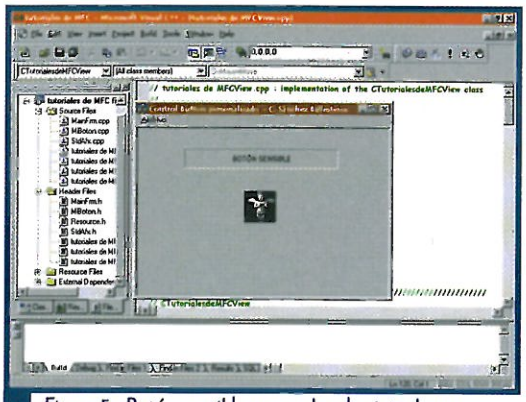


Figura 5.- Botón sensible agregado a la vista de una ventana.

Si nos fijamos bien en el listado anterior, vemos que se usa una función especial para poner el *bitmap* correspondiente en los límites que ocupa el botón (*Stretchblit*). *StretchBlt* amplía o comprime el *bitmap* origen en memoria, y copia el resultado en el rectángulo destino. Los datos de color para los *píxeles* destino son mezclados después de la ampliación o compresión de la imagen.

Stretchblt permite volcar un bitmap a un DC ajustándolo al tamaño deseado

Esta función permite realizar ciertas operaciones especiales con los *píxeles* origen y destino (operación *raster*). Si la operación *raster* que seleccionemos requiere un pincel, el sistema utilizará el que tenga seleccionado actualmente en el contexto de dispositivo (*DC*).

Las coordenadas destino son transformadas utilizando la transformación especificada para el *DC* destino. Igual ocurre para las coordenadas origen. Si el *DC* destino, fuente y los *bitmaps* no tienen el mismo formato de color, *StretchBlt* convierte el *bitmap* origen para que cuadre con el de destino.

StretchBlt puede crear una imagen "espejo" de un *bitmap* si el signo de los parámetros *nWidthSrc* y *nWidthDest* o *nHeightSrc* y *nHeightDest* difieren. Si *nWidthSrc* y *nWidthDest* tienen diferente signo, la función crea un espejo de la imagen en el eje *X*. Del mismo modo, *nHeight Src* y *nHeightDest* tienen diferentes signos, y la imagen espejo se creará en el eje *Y*.

En la *Tabla 1* podemos ver un detalle de los parámetros de esta función.

Para terminar el desarrollo de nuestro proyecto, sólo queda comentar el código referente a la creación del botón desde la vista de nuestro programa.

El mejor momento para crear el botón, es el de actualización de la vista (*OnInitialUpdate*). Ahí definiremos el tamaño físico del botón, lo crearemos y le asignaremos los *bitmaps* correspondientes mediante las funciones implementadas en la clase *CMiBoton*.

```
void
     CTutorialesdeMFCView::OnIniti
     alUpdate()
{
    CFormView::OnInitialUpdate();
    GetParentFrame()-
     >RecalcLayout();
    ResizeParentToFit();
    GetParent()->CenterWindow();

    //CREAR UN BOTÓN SENSIBLE
    CRect tamano;
    tamano.left =170;
    tamano.right =tamano.left+50 ;
    tamano.top=100;
    tamano.bottom = tamano.top+50;
    boton->Create ("", WS_VISIBLE
     | BS_OWNERDRAW ,
     tamano,this ,ID_MIBOTON);
     // Cargar bitmaps para el
      triestado
     boton->SelectBitmap (IDB_BIT-
      MAP1,IDB_BITMAP3,IDB_BIT-
      MAP2);
}

Para pasar constantemente la
     coordenada del ratón, utili-
     zaremos el evento OnMouseMove
     de la vista.

void
     CTutorialesdeMFCView::OnMouse
     Move(UINT nFlags, CPoint
     point)
{
    boton->TestButton(point);
    CFormView::OnMouseMove(nFlags,
     point);
}
```

Los controles owner-drawn permiten que personalicemos nuestros propios botones

Con esto tendremos nuestro botón sensible completamente funcional. En la Figura 5 podemos ver en tiempo de ejecución el proyecto desarrollado.

Terminamos esta serie, confiando en que os haya sido de utilidad. Para ampliar conocimientos sobre el desarrollo de programas *Windows* bajo *MFC's*, aconsejo la lectura de alguna publicación especializada. En mi opinión, uno de los libros más completos, es el escrito por *Francisco Javier Ceballos**, excelente complemento para los temas tratados en esta serie.

---

* Francisco Javier Cevallos. *Visual C++: Aplicaciones para Win32. RAMA.*

PREPARATE PARA SER EL MEJOR WEBMASTER

HASTA UN 45% DE DESCUENTO

- ✓ En España se ha duplicado el número de usuarios en los últimos 8 meses.
  Fuente: AIMC
- ✓ El 80% de internautas existentes en todo el mundo entran en la Red para realizar negocios.
  Fuente: FORRESTER RESEARCH
- ✓ 100 millones de personas navegan hoy.
  Fuente: Varios compilado por NUA INTERNET SURVEYS
- ✓ En 5 años el comercio electrónico supondrá 4,5 billones de pesetas tan sólo en los Estados Unidos.
  Fuente: IBID
- ✓ En el año 2.000 la gran mayoría de las transaciones no realizadas en dinero metálico serán mediante pagos electrónicos.
  Fuente: KILLEN ASSOCIATES

RECORTE Y ENVÍE EL CUPÓN ADJUNTO A:

Revistas Profesionales, S. L. • San Sotero, 5-1.ª planta • 28037 • Madrid • Fax: 91 327 13 03

BOLETÍN DE SUSCRIPCIÓN

OFERTA VÁLIDA SÓLO PARA ESPAÑA

SI, deseo suscribirme al **CURSO WEBMASTER DE PROGRAMACIÓN INTERNET**

CONDICIONES DE PAGO (MARQUE LA OPCIÓN DESEADA)

- ❑ 49 fascículos a **750 Ptas/u. TOTAL: 36.750 PTAS.**
- ❑ 4 carpetas para archivar fascículos a **950 Ptas./u. TOTAL: 3.800 PTAS.**
- ❑ 1 carpeta para archivar disquetes y CD's a **1.250 Ptas.**

VALORACIÓN TOTAL DE LA COLECCIÓN CON FRONTPAGE 2000 INCLUIDO: 71.500 PTAS.

- ❑ **Pago al contado.** Ahorro del **45%. TOTAL: 39.350 Ptas.**
- ❑ **Pago en 3 plazos.** Ahorro del **38%**. Cada plazo: **14.790 Ptas. TOTAL: 44.380**

ENTREGA DEL FRONTPAGE 2000 EN EL ÚLTIMO ENVÍO

Sin FrontPage 2000 valoración total: 41.800 Ptas.

- ❑ **Pago en 6 plazos** (con ahorro del **30%**): Cada plazo: **4.875 Ptas. TOTAL: 29.260**
- ❑ **Pago en 12 plazos** (con ahorro del **20%**): Cada plazo: **2.785 Ptas. TOTAL: 33.440**

FORMAS DE PAGO:

❑ Con **cargo a mi tarjeta VISA** Nº ____/____ /____ /____ Caduca .....

❑ **Domiciliación bancaria**

Ruego a Uds. que se sirva cargar en mi ❑ cuenta corriente ❑ libreta de ahorro el/los recibos que le serán presentados por REVISTAS PROFESIONALES, S.L. como pago de mi suscripción al **CURSO WEBMASTER DE PROGRAMACIÓN INTERNET**

CÓDIGO CUENTA CORRIENTE

ENTIDAD	OFICINA	DC	Nº CUENTA

❑ **Cheque** a nombre de REVISTAS PROFESIONALES, SL.

❑ **Contra reembolso** del importe más gastos de envío.

❑ **Giro Postal** (adjunto fotocopia del resguardo)*.

*Esta forma de pago es exclusivamente para pagos al contado

Firma

UTILICE MAYÚSCULAS PARA RELLENAR EL PEDIDO

Nombre y apellidos .....F. nacimiento.....

Domicilio ..... C.P. .....

Ciudad ..... Provincia .....Telf. .....

e-mail ..... Profesión .....

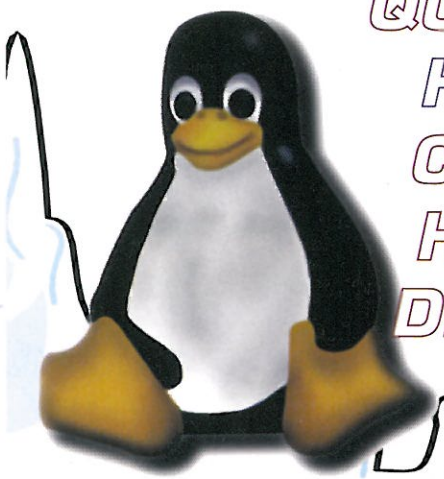
SUSCRIPCIÓN DOBLE

LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SÓLO PROGRAMADORES

Sólo PROGRAMADORES LINUX

QUE NO TE LÍEN CON EL <CÓDIGO>



PARA NO QUEDARSE HELADO CUANDO HABLEN DE LINUX

BOLETÍN DE SUSCRIPCIÓN

Rellene o fotocopie el cupón y envíelo a REVISTAS PROFESIONALES, S.L. (Revista Sólo Programadores).
C/ San Sotero, 5. 1ª Planta. 28037 Madrid. Tlf: 91 304 87 64. Fax: 91 327 13 03

Quiero suscribirme a la revistas SÓLO PROGRAMADORES y SÓLO PROGRAMADORES LINUX desde el Número .....
y beneficiarme de las condiciones de estas magníficas promociones:

❑ **SUSCRIPCIÓN ANUAL**
22 NÚMEROS + 22 CD-ROMs
AL PRECIO DE
14.822 ptas. / 89,09€

Precio de suscripción para el extranjero:
22 NÚMEROS + 22 CD-ROMs
AL PRECIO DE
21.200 Ptas.

FORMAS DE PAGO:

❑ Giro postal a nombre de REVISTAS PROFESIONALES, S.L
❑ Transferencia al Banco Popular Español. C/ Valdecanillas, 41.
Nº c/c: 0075/1040/43/ 0600047439
❑ Talón bancario a nombre de REVISTAS PROFESIONALES, S.L
❑ Domiciliación bancaria
❑ Contra reembolso

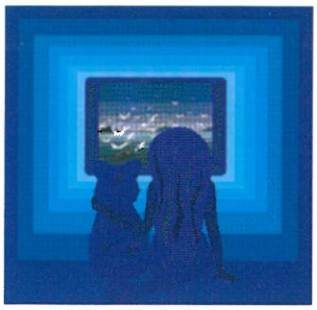
Soy antiguo suscriptor
❑ **Sí** ❑ **No**

PARA ENVÍOS AL EXTRANJERO SÓLO SE ADMITIRÁN LAS SIGUIENTES FORMAS DE PAGO:
❑ Giro postal a nombre de: REVISTAS PROFESIONALES, S.L
❑ Talón conformado en dólares a nombre de: REVISTAS PROFESIONALES, S.L
❑ Por VISA ____/____/____/____
Caducidad: __/__

NOMBRE Y APELLIDOS:.....
EDAD:..... PROFESIÓN: .....
TFNO: ..... DOMICILIO: .....
CIUDAD:..... C.P: .....PROVINCIA:.....

Datos de domiciliación:
Banco.....
Domicilio .....
Nº de Cuenta.....
Titular.....
Fecha .....
FIRMA

Promoción válida hasta agotar existencias



Programación OpenGL en Delphi (IV)

Juan Luis Ceada Ramos.
Diplomado en Informática de Sistemas.
Emilio Nestal Díaz.
Ingeniero Técnico en Informática.

En esta entrega aprenderemos a usar las primitivas relacionadas con los colores, las luces y las sombras que ayudan a dar realismo a las escenas.

COLORES

Desde la primera entrega de esta serie hemos venido usando las primitivas de selección de color, de tal forma que a estas alturas ha llegado el momento de profundizar un poco más en este aspecto. Existen varias formas de representar un color, pero lo más común en informática consiste en representar los colores mediante el espacio de color *RGB*.

Cada color se representa por una combinación de rojo (*R*), verde (*G*) y azul (*B*). El blanco está representado por el (*255,255,255*) y el negro por el (*0,0,0*).

Cada una de las posibles combinaciones existentes entre estas dos representan a un color diferente. Así, la combinación (*255,0,0*) representa al rojo y (*128,128,128*) a un gris de intensidad media.

Para seleccionar un determinado color en *OpenGl* disponemos de varias primitivas, siendo todas versiones de la función *glColor*. Se diferencian en el número de parámetros y en su tipo. A continuación veremos las principales.

En primer lugar tenemos *gl Color3f (r,g,b:real)* y *glColor4f (r ,g, b, alfa: real)*. En este caso, los componentes *RGB* son números flotantes y el rango de valores permitidos va de *0.0* a *1.0*. Por ejemplo, el color rojo sería (*1.0, 0.0, 0.0*). Los valores (*0.5, 0.5, 0.5*) representan a un gris de intensidad media. La versión de cuatro parámetros incluye además el canal *alfa*, que se utiliza para crear efectos de transparencia.

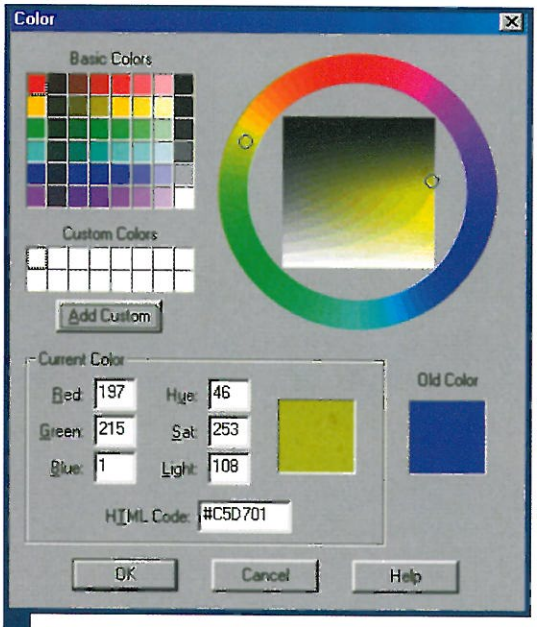


Figura 1.- Ventana de selección de colores representados por sus componentes *RGB*.

Normalmente, las dos funciones anteriores son las más usadas. Pero en el sistema *Windows* se suele trabajar con un rango de valores que va de *0* a *255*. Para facilitar la vida a los desarrolladores, *OpenGl* proporciona las funciones

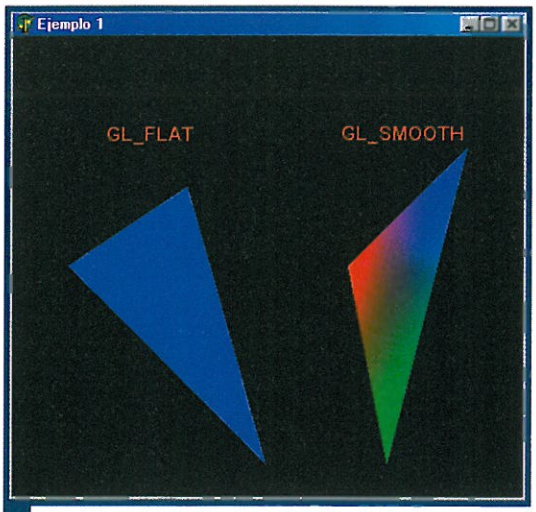


Figura 2.- Ejemplos de uso de *GL_FLAT* y *GL_SMOOTH*.

glColor3ub(r,g,b:byte) y *gl Color4ub (r ,g, b, alfa: byte)*. Estas funciones son totalmente equivalentes a las anteriores, pero ahorran tener que convertir los valores *RGB* de tipo *byte* con los que trabaja *Windows* a flotante.

Cuando establecemos un color, usando alguna de las funciones anteriores, realmente estamos indicando a *OpenGl* que desde este momento, dibuje todos los vértices con el color indicado. Por ejemplo, el siguiente código:

```
glbegin(GL_TRIANGLEs);
        glColor3ub(255,0,0);
        glvertex3i( 5, 0, 0);
        glvertex3i(0, 5, 0);
        glvertex3i(2,-2, 0);
glend;
```

dibujaría un triángulo de color rojo, ya que todos sus vértices son de ese color. En realidad, *OpenGL* ha rellenado el triángulo con un degradado de color, que tiene como color inicial el rojo, y como final también el rojo. Un degradado de un único color, da como resultado una superficie de un único color.

Sin embargo, el siguiente ejemplo produce un triángulo relleno por un degradado multicolor, puesto que *OpenGl* ha creado un degradado de color que parte del primer vértice (rojo), y va hacia el azul y el verde.

```
glbegin(GL_TRIANGLES);
      glColor3ub(255,0,0);
      glvertex3i( 5, 0, 0);
      glColor3ub(0,255,0);
      glvertex3i(0, 5, 0);
      glColor3ub(0,0,255);
      glvertex3i(2,-2, 0);
 glend;
```

Por supuesto, podemos elegir la forma en que se realizará este degradado de color. Para ello contamos con la función *glShadeModel(tipo)*, donde el parámetro *tipo* puede ser *GL_FLAT* (el polígono se rellena de un único color, por regla general, el del último vértice), o *GL_SMOOTH* (el polígono se rellena con un degradado de color). Con *GL _FLAT* el número de cálculos es menor (aumentando la velocidad de dibujado).

LUCES

En la vida real, los objetos, los colores, se ven porque existe luz. La luz condiciona el aspecto de los objetos. Tomemos como ejemplo la Figura 2: un fondo negro con dos triángulos con diferentes colores. Azules, rojos y verdes...bajo la luz del sol. Pero esa misma figura, en una habitación iluminada por una luz violeta, puede verse con tonos de color diferentes.

Básicamente hay tres tipos de luces: ambiente, especular y difusa. La luz ambiente, es aquella que no proviene de ningún punto en particular. La luz especular es direccional, es decir, proviene de un punto particular y se refleja igualmente de forma direccional. Un rayo láser puede ser considerado una luz especular casi perfecta. También el sol o un foco es una fuente de luz especular. La luz difusa es aquella que proviene de un punto, pero se refleja en múltiples direcciones (por ejemplo, una lámpara fluorescente).

Normalmente una escena de la vida real está iluminada por una mezcla de luces especulares, ambientales y difusas. De ahí que conseguir una escena generada por ordenador parezca real sea tan difícil, ya que hay que tener en cuenta cómo incide la luz sobre cada uno de los objetos, teniendo en cuenta además la forma, tamaño y los materiales.

Los materiales de los objetos condicionan el aspecto final

En las escenas con iluminación, realmente no creamos polígonos de un color u otro. En su lugar, creamos polígonos de un determinado material. Cada material tiene sus propias características de absorción y reflexión. Así, una esfera azul es de este color porque el material del que está hecha absorbe todas las longitudes de ondas excepto las que representan a los tonos azules.

Las fuentes de luz se definen mediante componentes *RGB*. Es decir, una luz de *(1.0,0.0,0.0)* es una luz roja. Supongamos que tenemos una luz ambiental con valor *(0.5,0.5,0.5)*. Con un polígono de un material con las siguientes características de reflectividad: *(0.5,0.0,1.0)*. Quiere decir que el polígono refleja la mitad de la luz roja que recibe, absorbe toda la verde, y refleja toda la azul. Si la luz ambiental incide sobre este polígono, este aparecerá ante nuestros

Figura 3.- Aspecto del mismo objeto bajo diferentes luces.

ojos con un color (*0.5*0.5, 0.0*0.5, 1.0*0.5*) = (*0.25,0.0,1.0*). Es decir, lo vemos de color azul (aunque con mezcla de rojo).

Si queremos ver un polígono de color rojo, tendremos que hacer que su material refleje por completo la luz roja: (*1.0,0.0,0.0*). Ahora creamos un foco de luz y dibujamos la escena. ¿El resultado es un polígono rojo?, depende del tipo de luz, de la distancia a la que se encuentre, etcétera. Si la luz es blanca y está situada encima del polígono, en el infinito, podemos decir que el polígono se verá de color rojo. Pero si la luz está más cerca, y situada en un ángulo de 45 grados, las zonas más alejadas del polígono no se verán de color rojo, sino que tendrán tonos cada vez más cercanos al negro. Ver Figura 4.

MATERIALES

Las características del material elegido condicionan el aspecto de los polígonos cuando hay fuentes de luz presentes. Hay materiales que reflejan muy bien la luz especular, pero que absorben la luz ambiental. También los hay que reflejan la luz ambiental, pero absorben la especular. Por ejemplo, una esfera metálica refleja la luz especular que incida sobre ella, provocando efectos como el que se puede observar en la Figura 5.

Cada vértice de nuestras primitivas tendrá un color *RGB* basado en la combinación de la luz ambiente, difusa y especular, multiplicado por la reflectancia ambiente, difusa y especular del material. Gracias al degradado que *OpenGl* introduce cuando los vértices de los polígonos son de distinto color, se produce la ilusión de que dichos polígonos se encuentran iluminados.

El proceso de combinar luces y materiales para obtener los efectos deseados no es algo matemático. Requiere de mucha práctica y experiencia. Es decir, el método de prueba y error llevado a sus últimas consecuencias. Como norma general podemos afirmar que las componentes ambientales y difusas de la luz son las que afectan más al color del objeto.

AÑADIR LUCES

Para activar el uso de luces tendremos que utilizar la función *glEnable(GL_LIGHTING)*. Esta función indica a *OpenGl* que debe realizar los cálculos necesarios para que los polígonos se vean afectados por las condiciones de luminosidad.

Mediante *glLight Model (tipo, valor)* indicamos cuál va a ser el tipo de luz que vamos a utilizar. El parámetro *tipo* admite varios valores, pero en esta entrega sólo usaremos *GL_LIGHT_MODEL_AMBIENT*, que indica que vamos a definir una luz ambiente global que ilumina a todos los objetos por todas sus caras. El parámetro *valor* es un *array* de cuatro posiciones que almacena el valor de las componentes *RGB* (más el canal *alpha*) de la luz (es decir, indican el color de esta última).

Hemos visto cómo los materiales de los objetos tienen mucho que ver con la iluminación. Dependiendo del material, los polígonos reflejaran más o menos luz. La forma más fácil de indicar el material es usando el método de *seguimiento de color*.

La idea es colocar una o más fuentes de luz, una serie de objetos, asignarles un material (con unas determinadas propiedades) y un color. Después, vamos cambiando el color de los objetos (mediante llamadas a *glColor*), y automáticamente, si el seguimiento de color está activado, *OpenGl* varía las características del material para que al incidir sobre él la luz se vea del color deseado. El siguiente código muestra como sería todo el proceso.

```
glEnable(GL_COLOR_MATERIAL);
glcolorMaterial(GL_FRONT,
    GL_AMBIENT_AND_DIFUSSE);
glMaterialfv(GL_FRONT,GL_AMBIENT_A
    ND_DIFFUSE, valores);
.....
glColorub(255,255,0);
.....
```

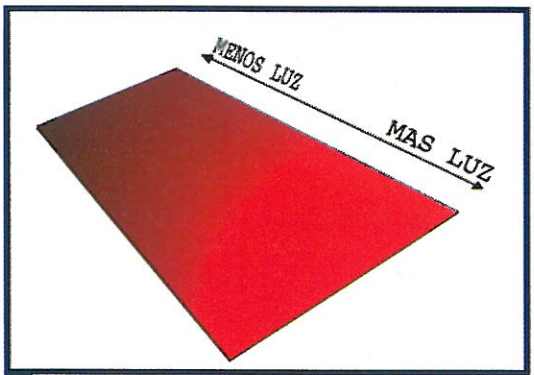


Figura 4.- Variación de color dependiendo de las características de la luz.

La primera línea activa el seguimiento de color. La segunda línea (función *glColorMaterial(cara, tipo)*) activa las propiedades ambiental y difusa del material de la cara frontal de los polígonos. Es decir, cuando cambiemos de color se va a modificar la reflectancia del material de la cara frontal del polígono, de forma que al incidir sobre él la luz, aparecerá con el color definido anteriormente.

La tercera línea (función *glMaterial(cara, tipo, valor)*) indica las características iniciales del material usado. En este caso, estamos definiendo las propiedades ambiental y difusa (las dos a la vez, por lo que tomarán los mismos valores) del material de la cara frontal de los polígonos. Por último, indicamos los valores de dichas propiedades (es decir, la reflectividad del material para ese tipo de luz), representado por una matriz de cuatro posiciones *RGBA*.

Más adelante, cambiamos de color (en este caso, el color elegido es el amarillo). *OpenGl* cambia automáticamente las propiedades de los materiales de la cara frontal de los polígonos, y a partir de ahora reflejarán más o menos las diferentes componentes de la luz, dibujándose los polígonos con el color deseado.

En la Tabla 2 encontraremos más información sobre los parámetros de cada una de estas funciones.

GENERACIÓN DE SOMBRAS

Dado que *OpenGL* no soporta la generación de sombras de forma nativa, es decir, no existe una primitiva que realice esta función, tendremos que realizarlas por nuestra cuenta. Pero antes de todo tendremos que tener en cuenta que es una sombra. Por sombra debemos entender la proyección cónica desde la posición de la luz de un objeto sobre un plano determinado, que es exactamente lo mismo que si decimos que una sombra no es mas que el objeto visto desde la posición de la luz y proyectado no sobre la pantalla, sino sobre uno o más de los planos de la escena. Recordemos anteriores entregas donde comentábamos el sistema de proyección de *OpenGL*.

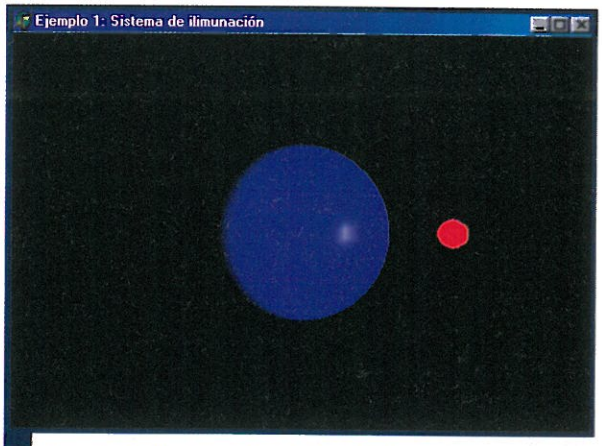


Figura 5.- Reflejos especulares (ejemplo 1).

En el sistema de proyección de *OpenGL*, veíamos cómo se creaba un cono limitado por la distancia máxima y por la distancia mínima, así como el ángulo de visión, donde se proyectaban en un plano los objetos que se encontraban en dicho cono. Este plano no es más que la pantalla de nuestro monitor y el foco del cono nuestra posición dentro de la escena.

Las sombras requieren unos sencillos conceptos matemáticos

Si entendemos esto, nos será fácil entender lo que debemos hacer para generar las sombras de los objetos que deseemos. Lo único que tenemos que hacer es construir una matriz de visualización que permita modificar este comportamiento estándar de *OpenGL*, y que permita cambiar el plano de la pantalla de nuestro monitor, por cualquier plano que pertenezca a la escena, además de modificar el foco del cono, que ahora será la posición del punto de donde procede la luz.

Para conseguir dicha modificación, tenemos que realizar una serie cálculos que implicarán unos sencillos conceptos matemáticos, concretamente algo de álgebra y de geometría. Primero hay que elegir el plano sobre el que se va a realizar dicha proyección, que vamos a definir mediante tres puntos que pertenezcan a dicho plano, y que será donde va a caer la sombra. Además, tendremos que elegir cuál es la posición de la luz que la va a generar. Con todo ello, utilizaremos una función que nos devolverá la matriz que tendremos que usar para modificar la estándar de *OpenGL*.

Dicha función tomará los tres puntos, creará dos vectores con ellos, y realizará el producto vectorial de ellos dos, es decir, una simple multiplicación. El resultado será un vector perpendicular al plano dado, el cual tendremos que normalizar, es decir, hacer que su módulo sea igual a uno (un par de sumas y una raíz cuadrada).

Una vez que tengamos el vector normalizado del plano que nos interesa, tendremos que realizar una serie de sencillos cálculos que

darán como resultado la matriz de transformación. Para mas detalles de la implementación de esta función, podéis consultar el código fuente del ejemplo número 2.

Hecho esto lo único que nos quedaría sería calcular tantas matrices de sombra como pares de luz-plano deseemos que intervengan en la generación de sombras, almacenarlas, y utilizarlas preferentemente mediante listas de visualización, cuando sea necesario.

Este método, permite además que tanto el plano como las luces se puedan mover en tiempo de ejecución, lo único que tenemos que hacer es construir la matriz cada vez que se modifiquen algunas de los parámetros que intervienen en su generación.

EN LA PRÁCTICA

En anteriores entregas hemos visto cómo realizar objetos, colocarlo en la escena, moverlos, rotarlos, etcétera. Ahora vamos a empezar con diferentes efectos que harán que dicha escena consiga tener un aspecto más real. El tema del color quedará expuesto de forma sumaria, ya que hemos visto lo suficiente en anteriores entregas. Profundizaremos en los sistemas de iluminación y como no, en la generación de sombras.

ALGO DE COLOR

La teoría del color es muy extensa y compleja y daría para muchas páginas, pero lo que realmente nos interesa del color es cómo utilizarlo en nuestros programas con *OpenGL*. Sabemos que en *OpenGL* podemos construir nuestras propias paletas, pero desde el punto de vista práctico no tiene mucho sentido debido a la capacidad de profundidad de color de las tarjetas actuales, 24 o 32 *bits*, y sólo en casos excepcionales tendría sentido la construcción de paletas personalizadas.

Por tanto, lo que de verdad nos interesa es cómo elegir el color que deseemos. Para ello, *OpenGL*, suministra la primitiva básica *glColorXY*, que ya conocéis. El hecho de tener multitud de funciones para el color evita la ardua tarea de convertir de un tipo a otro en tiempo de ejecución; así de simple, así de potente.

TABLA 2. Parámetros de *glColorMaterial* y *glMaterial*

Parámetro Cara	Descripción
GL_FRONT	Activar seguimiento de color para la cara frontal de los objetos.
GL_BACK	Activar seguimiento de color para la cara posterior de los objetos.
GL_FRONT_AND_BACK	Activar seguimiento de color para la cara frontal y posterior de los objetos.
Parámetro Tipo	**Descripción**
GL_AMBIENT, GL_DIFUSSE, GL_ESPECULAR, GL_AMBIENT_AND_DIFUSSE.	Indica las propiedades del material que se adaptan a los cambios de color. Si usamos *GL_DIFUSSE*, cuando cambiemos de color sólo cambiará la reflectividad del material ante luces difusas. La primitiva *GL_SHININESS* permite indicar un comportamiento del material, como si tuviera luz propia. Para terminar, debemos indicar los valores, en el caso de *GL_SHININESS* o bien el vector que los contiene, en el resto de los casos.

LUCES EN OPENGL

Hasta el momento, todos los ejemplos que hemos propuesto llevaban un sistema de iluminación básico. Este sistema básico permitía ver los objetos, sin efectos, ni reflejos. A partir de ahora, utilizaremos un sistema alternativo, algo más complejo, al cual añadiremos diferentes focos de luz y modificaremos las características de dichas luces. Tenemos que recordar que la mayoría de las implementaciones de *OpenGL*, entre ellas la de *Microsoft*, sólo soportan 8 focos de luz.

Para construir el sistema de iluminación lo primero que tenemos que hacer es activarlo, para lo que debemos utilizar *glEnable (GL _ LIGHTING)*. Así, conseguimos que *OpenGL* acepte las siguientes directivas sobre iluminación. A continuación tenemos que activar la luz que deseemos, mediante *gl Enable (GL_LIGHTX)*, donde la *X* especifica el número de la luz que deseamos activar. Antes de esto debemos dar las características a dicha luz. Para ello tenemos la primitiva *glLightfX(GL_LIGHTX, parámetro, valor_de_la_característica)*, donde parámetro puede ser una de las siguientes constantes: *GL_POSITION*, para establecer la posición de la luz, *GL_DIFFUSE* para especificar la componente difusa de la luz,

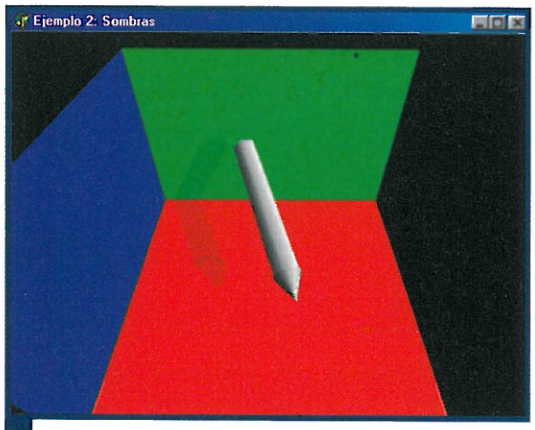


Figura 6.- Captura del ejemplo 2 (sombras).

GL_AMBIENT para la parte ambiental y *GL_SPECULAR*, para la parte especular.

Con ello, conseguimos la mitad de nuestro objetivo, y sabemos la importancia que tiene el material del objeto en el resultado final. Para completar la escena, hemos añadido características a los materiales de los diferentes objetos. Para ello hemos utilizado las siguientes primitivas *glColorMaterial(parte, tipo_luz)* y *glMaterialfX(parte, tipo_de_luz, valor)*.

El color de los objetos estará influido por los colores de las luces que incidan sobre los mismos

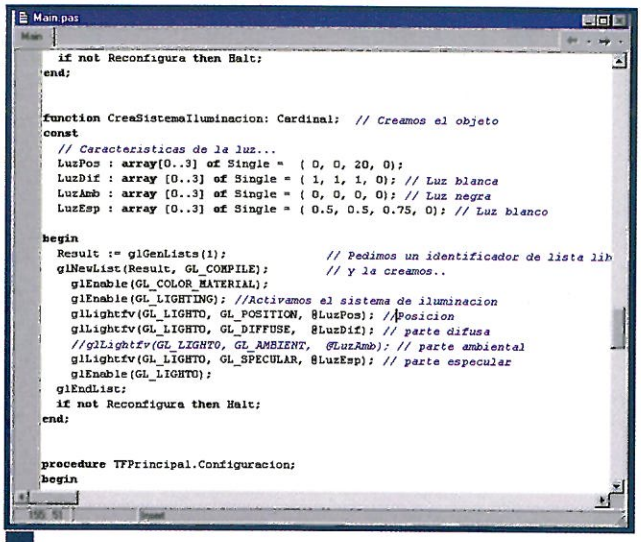
Hemos visto cómo podemos hacer que nuestros objetos se vean afectados por el sistema de iluminación. Pero sería interesante ver la forma de hacer lo contrario, es decir, que no se vean afectados, para conseguir en la misma escena, objetos afectados por la luz y otros no.

Para ello tendremos que desactivar el sistema de luz, dibujar aquellos que no deseemos que se vean afectados y volver a activar de nuevo el sistema de iluminación. Pero estaría bien si pudiéramos almacenar las características del sistema de luz completo. El siguiente código muestra cómo hacerlo de una forma fácil y sencilla.

```
glPushAttrib(GL_LIGHTING_BIT);
      glDisable(GL_LIGHTING)
       glCallList(LAtomo);
  glEnable(GL_LIGHTING)
     glPopAttrib();
```

Es tan fácil como almacenar en la pila de atributos el sistema de iluminación completo, hacer lo que deseemos, activar la luz de nuevo y sacar de la pila las características almacenadas.

Podéis comprobar cómo cambiando los diferentes parámetros que hemos visto, se pueden conseguir efectos bastante realistas, y sólo se deberán aplicar los conocimientos vistos en la parte teórica y un poco de sentido común.



```
Main.pas
  if not Reconfigura then Halt;
end;

function CreaSistemaIluminacion: Cardinal;  // Creamos el objeto
const
  // Caracteristicas de la luz...
  LuzPos : array[0..3] of Single =  ( 0, 0, 20, 0);
  LuzDif : array [0..3] of Single = ( 1, 1, 1, 0); // Luz blanca
  LuzAmb : array [0..3] of Single = ( 0, 0, 0, 0); // Luz negra
  LuzEsp : array [0..3] of Single = ( 0.5, 0.5, 0.75, 0); // Luz blanco

begin
  Result := glGenLists(1);              // Pedimos un identificador de lista lib
  glNewList(Result, GL_COMPILE);        // y la creamos..
    glEnable(GL_COLOR_MATERIAL);
    glEnable(GL_LIGHTING); //Activamos el sistema de iluminacion
    glLightfv(GL_LIGHT0, GL_POSITION, @LuzPos); //Posicion
    glLightfv(GL_LIGHT0, GL_DIFFUSE,  @LuzDif); // parte difusa
    //glLightfv(GL_LIGHT0, GL_AMBIENT,  @LuzAmb); // parte ambiental
    glLightfv(GL_LIGHT0, GL_SPECULAR, @LuzEsp); // parte especular
    glEnable(GL_LIGHT0);
  glEndList;
  if not Reconfigura then Halt;
end;

procedure TFPrincipal.Configuracion;
begin
```

Figura 6.- Código que crea las fuentes de luz en el ejemplo 1.

GENERACIÓN DE SOMBRAS

La generación de sombras es uno de los puntos débiles de *OpenGL*; no todo podía ser perfecto. *OpenGL* no soporta de forma nativa la creación de sombras, pero eso no va a ser problema, ya que utilizaremos un sistema de generación muy sencillo de entender y con unos resultados visuales impecables.

Para ello, hemos desarrollado unas funciones que generan la matriz necesaria para la generación de sombras. Lo primero que debemos hacer es elegir el plano sobre el que caerá la sombra y a continuación la posición de la luz que generará la sombra. Una vez que tengamos esos parámetros, utilizaremos la función *Construye Matriz DeSombra (vector _ puntos _ del _ plano, pos _ luz, m _ resultado)* que devolverá la matriz que debemos aplicar al sistema de proyección para conseguir la posición correcta para dibujar las sombras que necesitemos. Dicho método necesita un cálculo de matriz de proyección, por pareja de luz-plano, es decir, si la sombra puede proyectarse sobre dos planos, habría que hacerlo dos veces, lo mismo que si fueran dos luces.

Una vez calculada la matriz lo único que tenemos que hacer es utilizar la primitiva *glMultMatrixf(Matriz)*. Así conseguiremos que en la parte superior de la pila de matrices de proyección se coloque la que necesitamos, es decir, a partir de ese momento, todo lo que se dibuje aparecerá sobre el plano de proyección, apareciendo como sombras de lo que se dibuje.

Mediante glBlendFunc (...) conseguimos el efecto de transparencia

Esta función necesita de otras dos, *calculaNormal(vector_puntos_del_plano) y normaliza(vector)*, que se encargan de calcular la normal del plano que le pasamos a partir de tres puntos, y de normalizar dicho vector, respectivamente. Los aspectos técnicos de estas dos funciones son muy simples, la primera no es más que un producto vectorial de dos vectores y la segunda, no es más

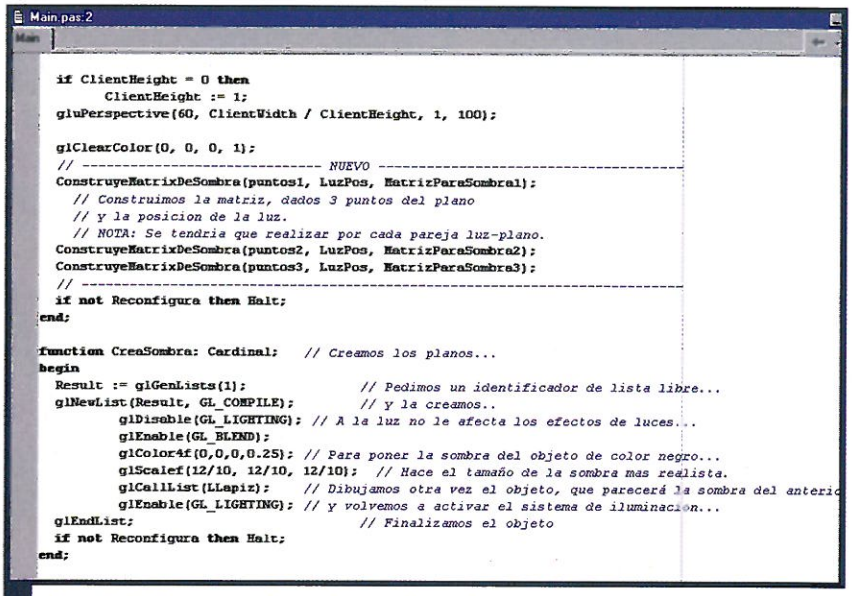

Main.pas:2
Main
if ClientHeight = 0 then
ClientHeight := 1;
gluPerspective(60, ClientWidth / ClientHeight, 1, 100);
glClearColor(0, 0, 0, 1);
// ---------- NUEVO ----------
ConstruyeMatrixDeSombra(puntos1, LuzPos, MatrizParaSombra1);
// Construimos la matriz, dados 3 puntos del plano
// y la posicion de la luz.
// NOTA: Se tendria que realizar por cada pareja luz-plano.
ConstruyeMatrixDeSombra(puntos2, LuzPos, MatrizParaSombra2);
ConstruyeMatrixDeSombra(puntos3, LuzPos, MatrizParaSombra3);
// ----------
if not Reconfigura then Halt;
end;
function CreaSombra: Cardinal; // Creamos los planos...
begin
Result := glGenLists(1); // Pedimos un identificador de lista libre...
glNewList(Result, GL_COMPILE); // y la creamos..
glDisable(GL_LIGHTING); // A la luz no le afecta los efectos de luces...
glEnable(GL_BLEND);
glColor4f(0,0,0,0.25); // Para poner la sombra del objeto de color negro...
glScalef(12/10, 12/10, 12/10); // Hace el tamaño de la sombra mas realista.
glCallList(LLapiz); // Dibujamos otra vez el objeto, que parecerá la sombra del anterio
glEnable(GL_LIGHTING); // y volvemos a activar el sistema de iluminacion...
glEndList; // Finalizamos el objeto
if not Reconfigura then Halt;
end;

Figura 7.- Código que crea las sombras en el ejemplo 2.

que realizar una suma y una raíz cuadrada.

Una vez que tenemos la matriz de visualización calculada, lo único que tenemos que hacer es dibujar aquello que deseemos que produzca sombra.

Para que la sombra tenga un aspecto más real tenemos que realizar una serie de modificaciones. Para empezar debemos desactivar la iluminación, con ello conseguimos que aparezca sin efectos de degradados, efectos de luz, etcétera También debemos dibujarla de color negro, añadiendo la capacidad de transparencia, mediante *glColor (0,0,0,0.25)*, donde la última cantidad se corresponde con el canal *alpha*.

OpenGL no soporta la generación de sombras de forma nativa, pero son fáciles de construir

Para lograr el efecto de transparencia debemos realizar dos cosas, primero decidir el tipo de función de transparencia, y después activarla con las siguientes primitivas de *OpenGL*: *gl Blend Func (GL _ SRC _ ALPHA, GL _ ONE _ MINUS _ SRC _ ALPHA)*, y *gl Enable (GL _ BLEND)*, respectivamente. La primitiva *glBlendFunc*(...) admite una serie de parámetros que en posteriores entregas veremos con detalle.

Como podemos imaginar el objeto-sombra aparecerá del mismo tamaño que el objeto que la produce. Pero como sabemos no es real, la sombra también depende de la distancia de la luz al objeto, cuando más cerca, más grande, y viceversa. Para conseguir este efecto, podemos calcular la distancia que separa al objeto de la luz, y calcular la proporción, para más tarde aplicársela a la primitiva *OpenGL*, *glScale(px, py, pz)*, que hará que el objeto-sombra se vea más grande o más pequeño.

Este método no es el más eficiente desde el punto de vista temporal, pero en cuanto a efecto conseguido no tiene comparación, ya que consigue sombras de realismo fotográfico.

CONCLUSIÓN

Queda probado cómo *OpenGL* tiene un soporte bastante considerable en cuanto a la iluminación, y aunque no soporta nativamente la generación de sombras, no ha sido nada difícil conseguir sombras en nuestros programas.

Se demuestra así otra de las características de *OpenGL*, la facilidad de implementación de otras posibilidades y funciones, y no resultará tan difícil que uniendo algo de lo que *OpenGL* aporta y un poco de nuestro ingenio, no podamos desarrollar lo que hallamos imaginado.

TABLA 3. Diferentes parámetros de *GlglLightfx*

GL_AMBIENT	Establece la componente ambiental de la luz. Vector de 4 valores.
GL_DIFFUSE	Establece la componente difusa de la luz. Vector de 4 valores.
GL_SPECULAR	Establece la componente especular de la luz. Vector de 4 valores.
GL_POSITION	Establece la posición de la fuente de luz. Vector de 4 valores.
GL_SPOT_DIRECTION	Establece la dirección del foco. Vector de 3 valores no normalizado.
GL_SPOT_EXPONENT	Un solo valor, que determina la intensidad de brillo del foco.
GL_SPOT_CUTOFF	Un solo valor, que especifica el ángulo de apertura del cono del foco de luz.
GL_CONSTANT_ATTENUATION	Valor de la atenuación constante de la luz.
GL_LINEAR_ATTENUATION	Valor de la atenuación lineal de la luz.
GL_QUADRATIC_ATTENUATION	Valor de la atenuación cuadrática de la luz.

Ahora puedes solucionarlo

Por fin los genuinos archivadores de SÓLO PROGRAMADORES

Podrás hacerte con ellos al precio de:

- Archivador de revistas: 900 Ptas.* (5,41€)
- Archivador de CD's: 700 Ptas.* (4,20€)
- Especial ahorro. Los dos archivadores por sólo 1.400 Ptas.* (8,41€)

Llama antes de que se agoten al teléfono 91 304 87 64, o escribe a la siguiente dirección:

BOLETÍN DE PEDIDO

Rellene o fotocopie el cupón y envíelo a REVISTAS PROFESIONALES, S.L.
(Revista Sólo Programadores Linux). C/ San Sotero, 5. 1ª Planta. 28037 Madrid
Tlf: 91 304 87 64. Fax: 91 327 13 03

Deseo que me envíen .....

.....

NOMBRE Y APELLIDOS: .....

.....

EDAD: ..... PROFESIÓN: ..... TFNO: .....

DOMICILIO: .....

CIUDAD: .....

PROVINCIA: ..... C.P: .....

Gastos de envío: 700 Ptas.

FORMAS DE PAGO:

❑ Giro postal a nombre de REVISTAS PROFESIONALES, S.L.
❑ Talón bancario a nombre de REVISTAS PROFESIONALES, S.L.
❑ Domiciliación bancaria
❑ Contra reembolso

DATOS DE DOMICILIACIÓN:

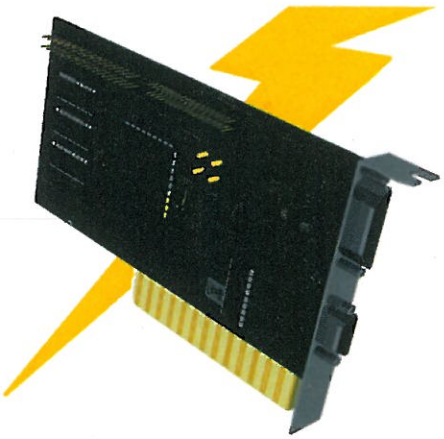
Banco ..... FIRMA

Domicilio .....

Nº de Cuenta ..... | ..... | ..... | .....

Titular .....

Fecha .....



Adquisición de datos con C++. Voltímetro y osciloscopio virtual

Félix Torán Marti y Diego Ramírez Muñoz.
Profesores investigaciones del Dpto. de Ingeniería Electrónica de la Universidad de Valencia.

La adquisición de datos y la instrumentación virtual abren las puertas del *PC* a la física, permitiendo recolectar información eléctrica para traducirla en medidas de temperatura, *PH*, etcétera. Nos valdremos para ello del desarrollo de un componente *VCL* con *C++ Builder*, aplicados para crear un voltímetro y un osciloscopio virtual.

La adquisición de datos y la instrumentación virtual abren las puertas del *PC* a la física, permitiendo recolectar información eléctrica para traducirla en medidas de temperatura, *PH*, etcétera. Nos valdremos para ello del desarrollo de un componente *VCL* con *C++ Builder*, aplicados para crear un voltímetro y un osciloscopio virtual.

Los dispositivos periféricos incorporados en un *PC* estándar no son suficientes para acceder a las variables del mundo físico que lo rodean. Podemos recibir y enviar información digital a través del puerto serie, pero..., ¿y si quisiéramos medir en tiempo real la temperatura de un líquido para mostrarla en pantalla y poderla monitorizar en remoto a través de *Internet*? Para ello deberemos insertar un nuevo periférico en nuestro ordenador: una tarjeta de adquisición de datos. En este artículo desarrollaremos un componente *VCL con C++ Builder*, que nos permitirá gobernar una tarjeta de adquisición de datos con suma facilidad. Finalmente, se ofrecerán ejemplos de programación que aplican y demuestran la potencia del componente.

TARJETA DE ADQUISICIÓN DE DATOS

Una tarjeta de adquisición de datos es un periférico capaz de transformar una o varias señales eléctricas analógicas en señales digitales. Al hacer la conversión, nuestros programas podrán conocer el valor de las tensiones eléctricas conectadas a la tarjeta. Señalaremos que, en general, una tarjeta de adquisición de datos incorpora los siguientes elementos (aunque no tienen por qué estar presentes todos):

- Un conjunto de líneas de entrada analógica. Estas permiten leer tensiones eléctricas analógicas para ser utilizadas por nuestros programas.
- Un conjunto de líneas de salida analógica. Permiten generar señales analógicas definidas por el programador.
- Un grupo de líneas de entrada digital. Permiten leer datos digitales.

- Un grupo de líneas de salida digital. Permiten escribir datos digitales.
- Otras líneas de utilidad variada (sincronización, generación de pulsos, etcétera.)

Las tensiones eléctricas conectadas a la tarjeta proceden de sondas, que convierten magnitudes físicas en magnitudes eléctricas

Muchas tarjetas presentan líneas digitales que pueden ser configuradas para lectura o escritura, según necesitemos en cada momento. También hay que saber que existe un gran número de parámetros que definen la calidad de nuestra tarjeta, como la diafonía y el factor de rechazo al modo común. Dicho esto, podemos asegurar que nuestros programas podrán leer niveles de tensión con suficiente precisión, sin tenernos que preocupar de nada más. Las tensiones eléctricas conectadas a la tarjeta suelen proceder de sondas, que convierten magnitudes físicas (con la temperatura, turbidez, etc.) en magnitudes eléctricas relacionadas con las anteriores (normalmente tensiones, aunque a veces se proporcionan corrientes).

CODIFICACIÓN DE LA INFORMACIÓN

Nuestra tarjeta nos proporciona palabras binarias que codifican tensiones eléctricas. Esta tarea la realiza el corazón del dispositivo: un convertidor analógico/digital (*CAD*). Admite un rango de tensiones de entrada (sea, por ejemplo, el rango de *0* a *10V*) fijado por el fabricante. El *CAD* divide el rango de entrada en un conjunto de niveles, que se numeran con palabras binarias consecutivas. A su salida proporciona la palabra binaria correspondiente al nivel más cercano a la tensión de entrada. Si llamamos *DV* al rango de entrada, y *n* es el número de *bits* del *CAD*, tendremos *n* niveles separados en una tensión $DV/(2^n-1)$.

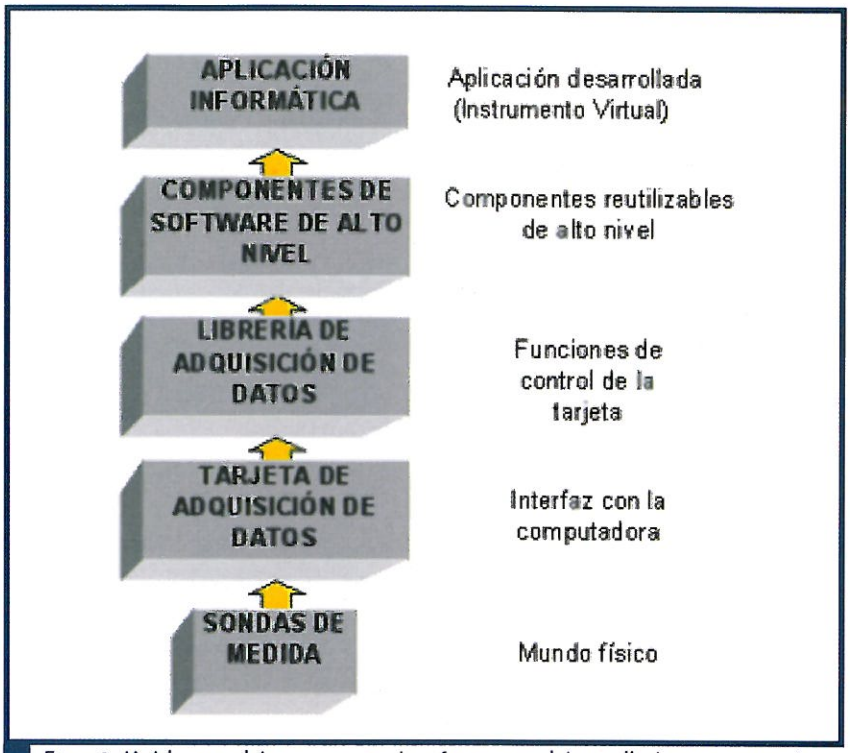

APLICACIÓN INFORMÁTICA
Aplicación desarrollada (Instrumento Virtual)
COMPONENTES DE SOFTWARE DE ALTO NIVEL
Componentes reutilizables de alto nivel
LIBRERÍA DE ADQUISICIÓN DE DATOS
Funciones de control de la tarjeta
TARJETA DE ADQUISICIÓN DE DATOS
Interfaz con la computadora
SONDAS DE MEDIDA
Mundo físico

Figura 1.- Modelo general de componentes de *software* para el desarrollo de instrumentación virtual *VI*.

Por ejemplo, un *CAD* de 12 *bits* con un rango de entrada en el intervalo *0...5 V* dividirá la tensión de entrada en $2^{12}=4096$ niveles de $5/(2^{12}-1)=1{,}221\ mV$ de tensión. Si introducimos en la entrada una tensión de *2* Voltios, el nivel más cercano será el numerado como 1638, luego convirtiendo a numeración binaria, la salida será *0110 0110 0110*, que será el valor leído por nuestros programas.

Como el lector habrá observado, los datos que proporciona la tarjeta no son directamente interpretables como tensiones, y es necesario aplicar una transformación matemática. En efecto, si leemos un valor decimal *D* y queremos conocer la tensión asociada (*V*), se debe aplicar la ecuación $V=D\cdot DV/(2^n-1)$.

Es fácil comprobar que en el ejemplo propuesto, se obtiene $V=1638\cdot 5/(2^{12}-1)=2\ Voltios$. El lector habrá observado que se produce un cierto error, puesto que hay que aproximar la tensión de entrada al nivel de cuantización más cercano. El error máximo equivale a la mitad de la tensión de paso entre niveles, que en nuestro ejemplo toma un valor de *0,61mV*, lo que es bastante aceptable. Por tanto, los *CAD* presentan mayor precisión (resolución) cuanto mayor sea el número de *bits* con que trabajan.

Como conclusión se debe retener que necesitamos conocer algunos datos sobre el CAD incorporado

en la tarjeta que utilicemos, para poder traducir los datos leídos a voltios. Si tenemos en cuenta que la mayoría de las tarjetas permiten aplicar una ganancia programable a las señales de entrada (para ajustar el rango de entrada), la transformación matemática se amplía más. Pero, ¿existe alguna forma más sencilla y transparente de hacer esto?

LIBRERÍAS DE ADQUISICIÓN DE DATOS

Las tarjetas más baratas, y sobre todo las más antiguas, tienen un manual que especifica la forma de configurarlas. La cantidad de operaciones posibles es elevada incluso para objetivos sencillos, y por ello la mayoría de programadores desarrollan sus propias librerías con funciones de alto nivel para el control de la tarjeta.

La mayoría de las tarjetas actuales proporcionan librerías de funciones, que encapsulan el complejo código necesario para realizar las tareas más comunes. Suelen venir en forma de librerías *DLL* (para enlazado dinámico), o ficheros *.LIB* (para enlazado estático). Incluso algunos fabricantes ponen una capa superior adicional, desarrollando componentes *ActiveX* que hacen aun más fácil el control de una o varias tarjetas de adquisición de datos, y permiten reutilizar dichos controles bajo muy diversos entornos de desarrollo. La Figura 1 resume todo lo explicado mediante un modelo general de componentes de *software* para el desarrollo de instrumentos virtuales, que encontraremos en la mayoría de los casos sin demasiadas variaciones.

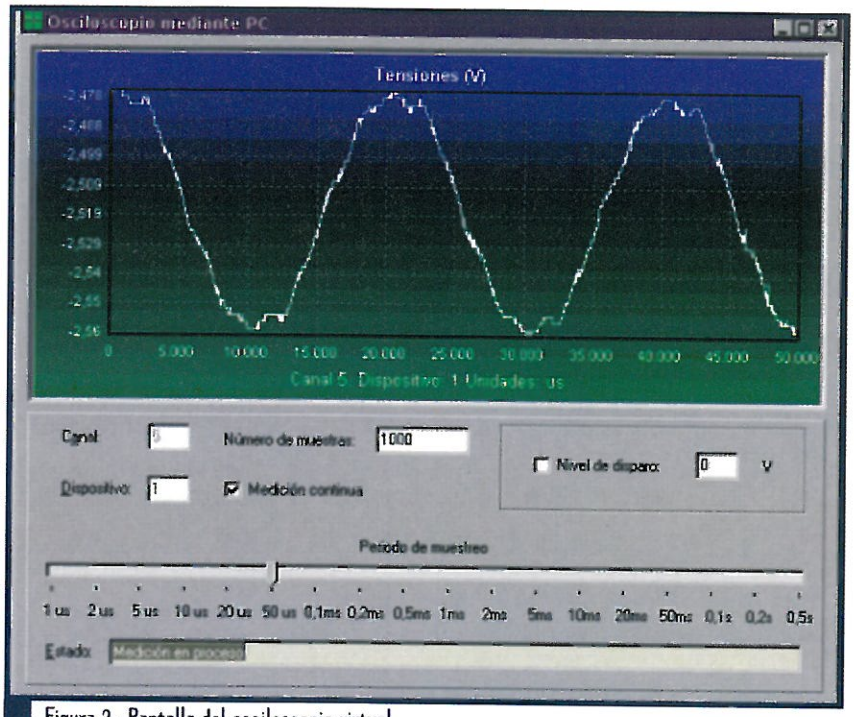


Figura 2.- Pantalla del osciloscopio virtual.

Hay lenguajes de programación gráfica (como *LabVIEW*) que sustituyen a la capa de componentes de *software,* y permiten un rápido y eficaz desarrollo de *VIs* (*Virtual Instruments*).

ETAPA DE DISEÑO

Vamos a situarnos en las capas superiores del modelo, desarrollando con *C++ Builder* un componente *VCL* reutilizable que facilite algunas de las tareas típicas de adquisición de datos. Los códigos fuente que acompañan al artículo (ver *CD*)se han desarrollado con la versión *3* de *C++ Builder*, aunque pueden ser abiertos y compilados con la versión *4*. Utilizando las posibilidades de *Borland*, el lector podrá convertir *VCL* en un control *ActiveX*, para utilizarlo en documentos *HTML*, *Visual Basic*, *Visual C++*, *Visual J++*, *LabVIEW 5*, y un sinfín de productos. Invitamos al lector a utilizar conceptos de programación orientada a objetos, y crear un componente derivado que implemente nuevas funcionalidades.

Los códigos fuente que acompañan al artículo se han desarrollado con la versión 3 de C++ Builder

No podemos abandonar las capas inferiores, luego es necesario seleccionar una tarjeta de adquisición de datos (un fabricante) y una librería de funciones diseñada para la tarjeta seleccionada. Vamos a utilizar la librería *NI-DAQ* de la firma *National Instruments*, que es gratuita y soporta la mayoría las de tarjetas distribuidas por esa compañía. Hay mucha información técnica en *Internet*, en la dirección http://www.natinst.com. El *software* de instalación de *NI-DAQ* se puede obtener gratuitamente en la dirección *ftp:// ftp.natinst.com/*

support/ daq/pc/ni-daq/. El paquete incorpora varias herramientas y mucha documentación de interés para el programador. También se puede recurrir a otros muchos fabricantes de tarjetas de gran calidad (por ejemplo, *Keithley*), pudiendo obtener las correspondientes librerías de funciones en la mayoría de los casos.

ESTUDIO PREVIO

Antes de comenzar a escribir el código fuente del componente, debemos estudiar las funciones integradas en la librería *NI-DAQ* y seleccionar las que nos van a hacer falta. Se distribuye en forma de *DLL* y también como fichero *.lib* para diversos entornos de desarrollo. En la Tabla 1 se describen las funciones que vamos a utilizar (se pueden consultar todas ellas en la documentación de *NI-DAQ*). En la Tabla 2 se comenta el significado de los parámetros tomados por éstas. Con la elección que se ha realizado, el lector ya puede intuir la funcionalidad que va a tener el componente.

DEFINICIÓN DEL COMPONENTE

Ha llegado el momento de especificar la funcionalidad que vamos a dar a nuestro componente. Éste implementará las siguientes propiedades:

- *Dispositivo*. Valor entero corto que indica el dispositivo sobre el que se aplicarán los diversos métodos del componente. Es muy útil si se decide controlar varias tarjetas instaladas en el sistema.
- *Canal*. Valor entero corto que indica el canal analógico de entrada al que deben dirigirse las tareas indicadas por los métodos del componente.
- *NumeroMuestras*. Valor entero corto que indica el número de muestras a adquirir en la operación de adquisición de varios datos.
- *FrecMuestreo*. Valor de tipo *double* que indica la frecuencia de muestreo a aplicar en las tareas de adquisición (en muestras por segundo). Este valor está limitado por la tarjeta utilizada.

Respecto a los métodos, se implementarán los siguientes:

- *void __fastcall Configurar (TModo modo, short rango, T Polaridad polaridad)*. Permite configurar un canal analógico de entrada (indicado mediante la propiedad *Canal*). Los argumentos permiten configurar el modo de entrada, rango y polaridad del canal seleccionado. El valor del argumento *rango* debe ser consultado para cada tarjeta, ya que cada una ofrece unos rangos de entrada diferentes.
- *void __fastcall Reset( )*. Inicia el *hardware* de adquisición de datos.
- *void __fastcall AdquirirDatos (double *tensiones)*. Inicia la adquisición de tantos datos como indique la propiedad *NumeroMuestras*, aplicando la frecuencia de muestreo indicada en la propiedad *FrecMuestreo*. El proceso es sincrónico, de forma que el

TABLA 1. Funciones de adquisición de datos a utilizar en el desarollo del componente.

`short AI_Configure (deviceNumber, chan, inputMode, inputRange, polarity, driveAIS)`	Configura un canal de entrada analógico.
`short DAQ_Clear (deviceNumber)`	Inicializa el hardware de adquisición de datos.
`short DAQ_Op (deviceNumber, chan, gain, buffer, count, sampleRate)`	Adquisición síncrona de varios datos, que se devuelven a través del puntero "buffer" sin convertir a tensiones.
`short DAQ_VScale (deviceNumber, chan, gain, gainAdjust, offset, count, binArray, voltArray)`	Convierte un array de datos binarios a las correspondientes tensiones.
`short AI_VRead (deviceNumber, chan, gain, voltage)`	Lectura de tensión de un canal analógico.

TABLA 2. Parámetros utilizados por las funciones de adquisición de datos

deviceNumber	*short*	Dispositivo al que hace referencia una función. La numeración la asigna el administrador de dispositivos NI-DAQ.
chan	*short*	Canal al que hace referencia una función de adquisición de datos.
inputMode	*short*	Modo de entrada (0=diferencial, 1=tensión referida a masa, 2=flotante)
inputRange	*short*	Rango de tensiones de entrada. Depende de la tarjeta, y de la polaridad seleccionada.
polarity	*short*	Polaridad (0=bipolar, 1=unipolar: solo tensiones positivas)
driveAIS	*short*	Para medidas flotantes, y solo con algunas tarjetas. Normalmente, se utilizará valor 0.
gain	*short*	Ganancia a aplicar. El valor usual es 1.
buffer	*short*	Puntero a un buffer en el que se almacenan los resultados de algunas operaciones.
gainAdjust	*double*	Factor de ajuste de ganancia. El valor usual es 1.
offset	*double*	Desplazamiento a aplicar a los valores adquiridos.
binArray	*short*	Array de datos binarios adquiridos.
voltArray	*double*	Array de tensiones convertidas a partir de datos binarios.
count	*int*	Número de muestras a adquirir.
sampleRate	*double*	Frecuencia de muestreo
voltage	*double*	Tensión leída.
status	*short*	Código de estado devuelto por todas las funciones de la librería. Toma valor nulo si la operación se ha completado con éxito. Si el valor es positivo, la función se ha ejecutado pero puede ocasionar problemas. Si el valor es negativo, corresponde a un código de error.

método devuelve el control cuando todos los datos han sido adquiridos. El resultado se devuelve en el *array* apuntado por el argumento *tensiones*, conteniendo los valores de tensión leídos en Voltios.

- *double __fastcall* Adquirir Dato ( ). Realiza la adquisición de un dato, devolviendo la tensión leída en Voltios, como valor de doble precisión.

Tan solo incorporaremos dos eventos:

- *On Error (TObject *Sender, short Status)*. Evento de notificación de errores. Se proporciona el código de error producido (consultar en la documentación de *NI-DAQ*).

- *OnErrorLeve(TObject *Sender)*. Notifica que la última operación realizada ha concluido correctamente, pero puede producir efectos secundarios.

Como tipos de datos, definimos dos enumeraciones:

- *TModo*. Enumeración que especifica el modo de entrada para un canal. Admite los valores *mDiferential* (modo diferencial), *mRSE* (referido a masa) o *mNRSE* (flotante).

- *TPolaridad*. Enumeración que indica la polaridad de un canal de entrada. Admite los valores *pUnipolar* (tensiones positivas) y *pBipolar* (tensiones positivas y negativas).

PROGRAMACIÓN DEL ARCHIVO DE CABECERA

Ya se ha especificado por completo el componente, y por tanto podemos empezar a escribir el código fuente. Después de abrir *C++ Builder*, creamos un nuevo paquete de componentes, llamado *PackAdquisicion.bpk* utilizando la opción *New* del menú *File*. Después, añadimos el esqueleto del nuevo

componente con la opción *Add to project* del menú *Project*, seleccionando pestaña *New Component*. A continuación, seleccionamos *TComponent* como clase base (*Ancestor type*) y *Tadquisicion Datos* como nombre del componente (*ClassName*). También indicaremos el nombre de una pestaña de componentes existente o teclearemos el nombre de una nueva (por ejemplo, *Adquisicion*). Finalmente, damos una ruta y un nombre para el archivo de implementación de la nueva clase: *AdquisicionDatos.cpp*. Sólo queda guardar todo para que se almacenen en disco los archivos creados (opción *Save All*).

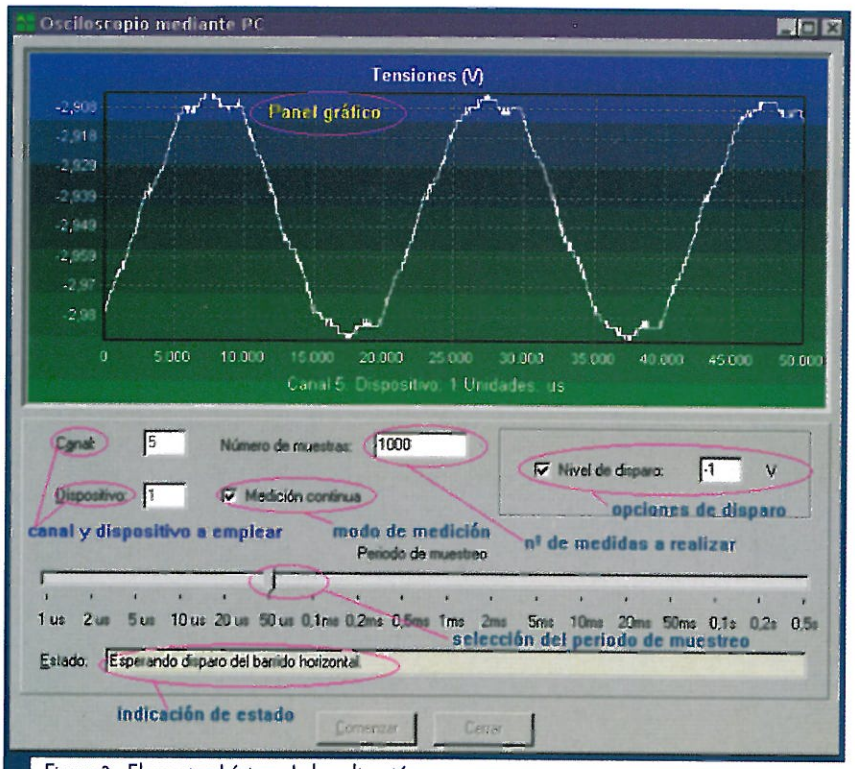


Figura 3.- Elementos básicos de la aplicación.

Al utilizar enlazado estático, no será necesario distribuir librerías DLL junto a las aplicaciones

Ahora hay que abrir el editor de código y mostrar el archivo de cabecera, con la opción *Open source/header file*. Es el momento de declarar las propiedades, métodos, eventos y tipos de datos del componente, usando el código mostrado en el Listado 1 (incluido en el código fuente del *CD*).

En nuestro ejemplo vamos a usar el enlazado estático con la librería *NI-DAQ*. Copiaremos primero el archivo *..\lib\nidaq32b.lib* proporcionado por *NI-DAQ* al directorio de nuestro código fuente. Es más usual copiar dicho fichero al directorio cccc de *C++ Builder* para que todos los proyectos puedan usar la librería, aunque también se puede hacer en otro directorio, con la condición de indicar su situación en la configuración de directorios (menú *Project -> options*). Terminamos editando el código fuente de proyecto del paquete de componentes (*Pack Adquisicion. cpp*) e incluyendo la línea *USELIB (nidaq32b.lib);*

Así indicamos al compilador que vamos a utilizar enlazado estático, y por tanto no será necesario distribuir librerías *DLL* junto a las aplicaciones que usen nuestro componente, ganando también en velocidad de ejecución (ya que se evita el enlace dinámico). El inconveniente es que las aplicaciones ocuparán mayor espacio en disco.

PROGRAMACIÓN DEL ARCHIVO DE IMPLEMENTACIÓN

Ya definida la interfaz de nuestro nuevo componente debemos escribir el código fuente que le dará funcionalidad. Ya que vamos a utilizar algunas de las funciones de la librería *NI-DAQ*, el compilador deberá conocer el prototipo de éstas, que se encuentra en el archivo *..\include\nidaq.h* situado en el directorio de instalación de la librería *NI-DAQ*. Copiamos dicho fichero, preferiblemente al directorio *include* de *C++ Builder*, e incluimos la línea *#include "nidaq.h"* en el archivo de implementación del componente (*Adquisicion Datos.cpp*). El código de implementación se presenta comentado con detalle en el *CD-ROM*. Su funcionamiento se puede resumir con los siguientes métodos:

- *Constructor*: inicia los valores por defecto de las propiedades.
- *Método Configurar*: se llama a la función *AI_Configure* de la librería *NI-DAQ*, pasando los parámetros alojados en las propiedades. Así se configura un canal con el modo, rango y polaridad deseados.
- *Método Reset*: se llama a la función *AI_Clear* de *NI-DAQ*, consiguiendo reiniciar el *hardware*.

LISTADO 1. Declaración de propiedades, métodos, eventos y tipos de datos

```
//
#ifndef AdquisicionDatosH
#define AdquisicionDatosH
//
#include <SysUtils.hpp>
#include <Controls.hpp>
#include <Classes.hpp>
#include <Forms.hpp>
//
//Tipos de datos
typedef enum {mDiferential, mRSE, mNRSE} TModo;
typedef enum {pUnipolar,pBipolar} TPolaridad;

//
class PACKAGE TAdquisicionDatos : public TComponent
{
  private:
  //Miembros de propiedades
  short FDispositivo, FCanal, FNumeroMuestras;
  double FFrecMuestreo;
  //Miembro para evento OnError
  typedef void __fastcall (__closure *TOnErrorEvent)(TObject *Sender,short Status);
  TOnErrorEvent FOnError;
  //Miembro para el evento OnErrorLeve
  TNotifyEvent FOnErrorLeve;
  //Lanza el evento de error apropiado, en función del resultado de una función
  //de la librería NI-DAQ (Status)
  void __fastcall TestError(short);

public:
   __fastcall TAdquisicionDatos(TComponent* Owner);

   //Configuración de canales analógicos
   void __fastcall Configurar(TModo, short, TPolaridad);
   //Inicialización del hardware
   void __fastcall Reset();
   //Adquisición múltiple y síncrona de tensiones.
   void __fastcall AdquirirDatos(double *);
   //Aquisición de una tensión.
   double __fastcall AdquirirDato();

  __published:
   //Declaración de propiedades

   __property short Dispositivo={read=FDispositivo,write=FDispositivo,default=1};
   __property short Canal={read=FCanal,write=FCanal,default=0};
   __property short NumeroMuestras={read=FNumeroMuestras,write=FNumeroMuestras,default=100};
   __property double FrecMuestreo={read=FFrecMuestreo,write=FFrecMuestreo,default=1000};

   //Declaración de eventos
   __property TOnErrorEvent OnError={read=FOnError,write=FOnError};
   __property TNotifyEvent OnErrorLeve={read=FOnErrorLeve,write=FOnErrorLeve};
};
//
#endif
```

- *Método TestError (privado)*: recibe un código de error de *NI-DAQ*, y lanza uno de los eventos de error en función de la gravedad: si el error es grave se lanza el evento *OnError*, mientras que si es leve se lanza *OnErrorLeve*.

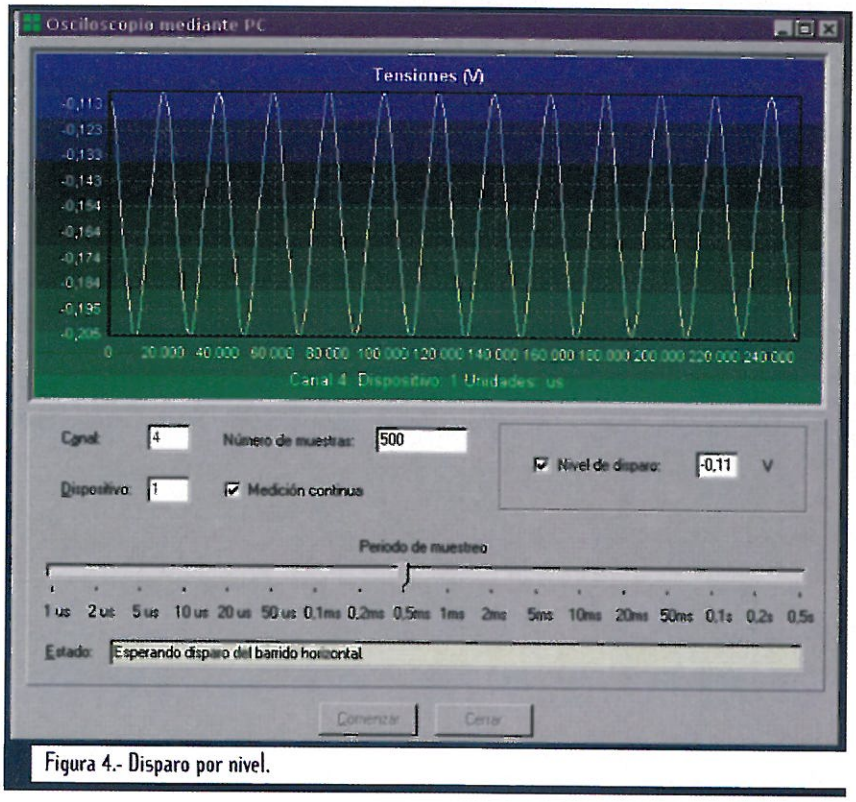
- *Método AdquirirDato*: aplica la función *AI_Vread* para adquirir una lectura puntual de tensión en el canal activo, devolviendo su valor.

- *Método AdquirirDatos*: utiliza la función *DAQ_Op* para efectuar un barrido de adquisición de datos y almacenar los datos leídos en un *array* (en formato binario). Acto seguido, aplica la función *DAQ_VScale* para convertir los datos binarios en tensiones, que se almacenan en el *buffer* definido por un puntero pasado como argumento.

Antes de concluir cada método, se hace una comprobación de errores, lanzando el evento apropiado según la gravedad del error producido. Esto se hace llamando al método *TestError*, y pasando como parámetro el valor de estado que devuelven las funciones de adquisición. Hay que recordar que un valor positivo indica error leve, mientras que un valor negativo apunta a un error grave.

El código de implementación se presenta comentado con detalle en el CD-ROM

El último paso sería usar las potentes opciones de *C++ Builder* para crear un control *ActiveX* a partir de nuestro nuevo componente. De esta forma, se podrían añadir funciones de adquisición de datos a los programas desarrollados con multitud de lenguajes y entornos de desarrollo.


Osciloscopio mediante PC
Tensiones (V)
Canal 4 Dispositivo: 1 Unidades: us
Canal: 4
Número de muestras: 500
Dispositivo: 1
Medición continua
Nivel de disparo: -0,11 V
Periodo de muestreo
1 us 2 us 5 us 10 us 20 us 50 us 0,1ms 0,2ms 0,5ms 1ms 2ms 5ms 10ms 20ms 50ms 0,1s 0,2s 0,5s
Estado: Esperando disparo del barrido horizontal
Comenzar
Cerrar
Figura 4.- Disparo por nivel.

VOLTÍMETRO Y OSCILOSCOPIO VIRTUAL

Como aplicación final del componente desarrollado, y como prueba de lo sencillo y rápido que resulta para crear instrumentos virtuales, se han desarrollado 2 aplicaciones de ejemplo:

- Voltímetro Virtual: permite leer (incluso en continuo) tensiones de hasta 8 canales analógicos de entrada simultáneamente.

- Osciloscopio virtual: (Figura 3) permite visualizar la evolución temporal de señales de tensión, utilizando cualquiera de los canales analógicos disponibles. Se comporta de forma similar a un osciloscopio de laboratorio. Puede funcionar en modo barrido o continuo, y utilizar disparo por nivel si se desea.

La licencia y el código fuente de ambas aplicaciones (*para C++ Builder 3*) se presenta exhaustivamente comentado, y se puede hallar en el *CDRom*

MANUAL DEL OSCILOSCOPIO

La aplicación *Osciloscopio.exe* es un instrumento virtual (*Virtual Instrument*, o *VI*) de gran utilidad en un laboratorio de electrónica (e incluso en otros campos de la cien-

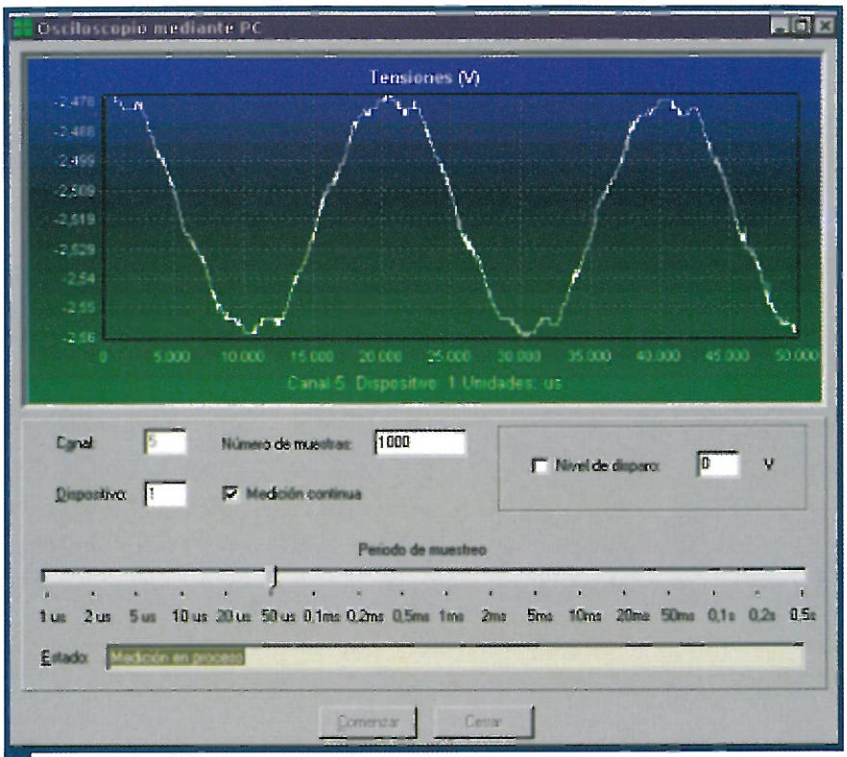


Figura 5.- Señal DC negativa con ruido.

cia, como la física, química, etcétera), ya que permite visualizar en pantalla la evolución temporal de señales de tensión eléctrica, que pueden estar relacionadas con otras magnitudes físicas (como la temperatura, el pH, la presión atmosférica, etcétera). El hecho de utilizar una computadora y una tarjeta de adquisición de datos introduce grandes ventajas, destacando algunas de ellas a continuación:

El osciloscopio virtual es mucho más barato que los convencionales, y tiene mejores prestaciones

- Cambiando tan solo el *software* (ejecutando otro programa de instrumentación virtual) el sistema se convierte automáticamente en otro equipo de medida y automatización. Esto abarata el sistema, ya que con un solo ordenador se puede hacer el papel de varios equipos de laboratorio.

- En el caso concreto del osciloscopio virtual, existe la ventaja de disponer de bastantes entradas de señal (dependiendo de la tarjeta utilizada pueden ser incluso 64), al contrario de lo que ocurre en la mayoría de los osciloscopios comerciales, donde solo se dispone de 2 entradas. Aunque esta aplicación solo es capaz de monitorizar una señal cada vez, el usuario puede ampliar el funcionamiento del programa para visualizar varias señales simultáneamente (ya que se proporciona el código fuente completo para compilar con *Borland C++ Builder 3*).

- Con una buena elección de la tarjeta de adquisición a utilizar, se puede ahorrar una considerable cantidad de dinero, en comparación con el precio actual de un osciloscopio.

Un solo ordenador puede hacer la función de varios equipos de laboratorio

- Durante el desarrollo de un *VI* de cualquier tipo, se puede instalar en la computadora el programa *Osciloscopio*, pudiendo así monitorizar las señales que están siendo adquiridas por cada entrada analógica de la tarjeta de adquisición de datos, e incluso en algunos puntos concretos de la circuitería eléctrica. En dichas situaciones, el programa se convierte en una potente y rápida herramienta de *test* y ahorra mucho trabajo. En otras palabras, el osciloscopio está virtualmente integrado en el sistema en desarrollo.

CARACTERÍSTICAS TÉCNICAS

A continuación se detallan las características técnicas del Instrumento Virtual y algunas anotaciones importantes:

- Las especificaciones técnicas referentes a la electrónica de adquisición de datos (como el rango de entrada, máxima frecuencia de muestreo, etcétera) dependen de la tarjeta de adquisición de datos empleada. Será necesario consultar la documentación proporcionada por el fabricante y respetar las advertencias indicadas por el mismo.

- El componente de adquisición de datos utilizado por el programa emplea la librería *NI-DAQ* de *National Instruments*, por lo

que sólo se podrán utilizar las tarjetas de dicha firma soportadas por *NI-DAQ*. Editando el código fuente de la aplicación se pueden modificar fácilmente los componentes de adquisición de datos para dar soporte a otros tipos de tarjetas (la mayoría de fabricantes proporcionan librerías específicas para el gobierno de sus tarjetas, y nos sitúan en un nivel suficientemente alto para ampliar cómodamente el componente).

- El programa ha sido desarrollado y probado con la versión *3* de *C++ Builder*, por lo cual se recomienda compilarlo en dicho entorno. No debería haber problemas si se utiliza la versión *4*, pero no ha sido probado.

- El programa presenta la posibilidad de disparar el barrido horizontal por nivel, lo que tiene gran utilidad en la mayoría de las situaciones prácticas, como se verá más adelante en este mismo texto.
- El número de puntos que adquirir es programable de forma rápida.

- La frecuencia de muestreo que aplicar también es configurable.

- El proceso de medición puede funcionar en continuo o mediante un único barrido. Por tanto, se podrán monitorizar tanto señales periódicas como transitorias.

- La presentación de señales en la pantalla se realiza a través de un panel gráfico, que auto-ajusta el eje vertical y etiqueta el eje horizontal en unidades de tiempo, consiguiendo una cómoda visión e interpretación de las señales adquiridas.

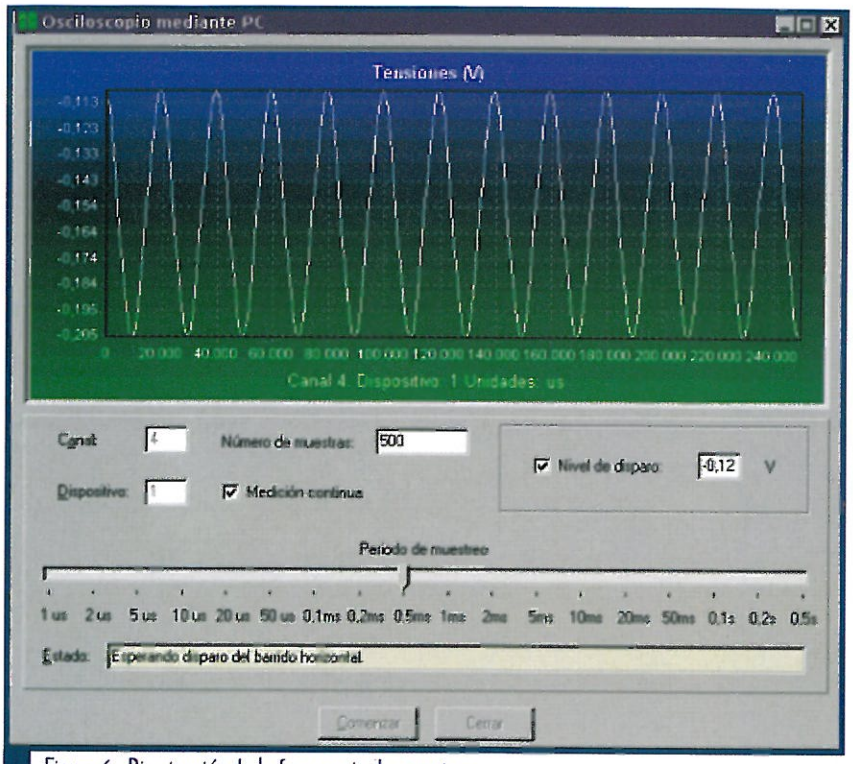


Figura 6.- Disminución de la frecuencia de muestreo.

El programa ha sido probado con una tarjeta *AT-MIO-16DE-10* de la firma *National Instruments*, que permite una frecuencia de muestreo máxima de 100*KHz*, con lo que se han podido visualizar con claridad señales de hasta 10 *KHz*, con un rango disponible de *[0V..10V]* o *[-5V..+5V]*. Empleando tarjetas más potentes, se pueden conseguir mayores anchos de banda.

Antes de concluir cada método, se hace una comprobación de errores

Este programa se presenta como un ejemplo de aplicación de componentes *VCL* de adquisición de datos, con fines didácticos. Por tanto, es una versión reducida de lo que puede llegar a ser un osciloscopio virtual. El usuario es libre de modificar el código fuente para ampliarlo y ajustarlo a sus necesidades (véase el apartado 1.8 "licencia de uso"), siendo posible conseguir una herramienta de valor incalculable en las tareas de *test* y medida.

El usuario es libre de modificar el código fuente

La Figura 1 muestra los elementos básicos que componen la interfaz de usuario, que se comentan con detalle en los siguientes apartados.

PANEL GRÁFICO

El panel gráfico muestra los valores de tensión medidos en función del tiempo. El eje vertical de la gráfica indica los valores de tensión, ajustándose automáticamente al rango necesario para que la señal sea vista en su totalidad. El eje horizontal se etiqueta automática-

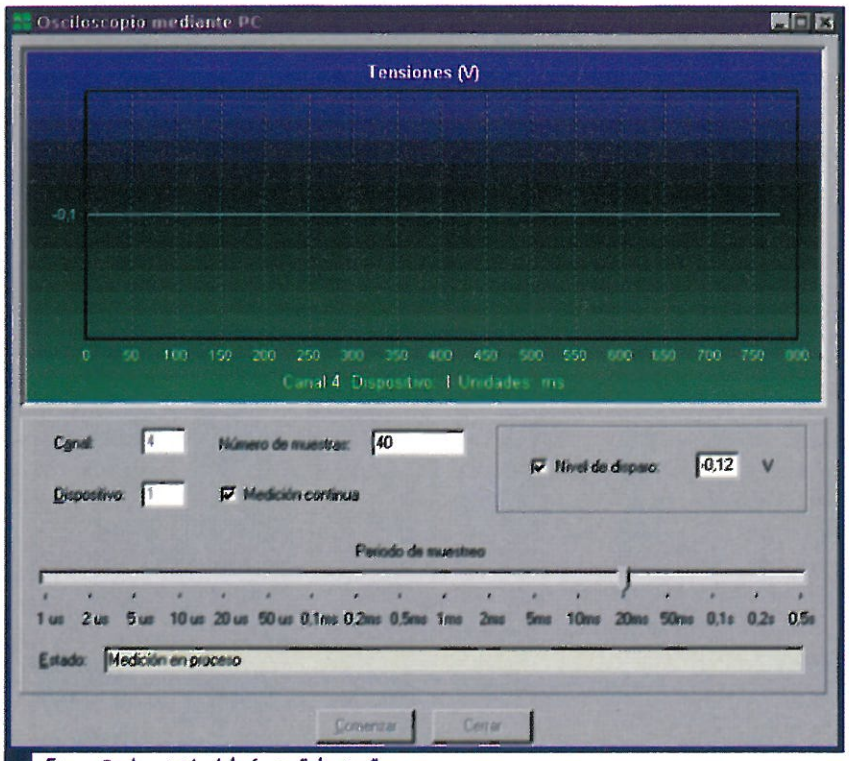

Osciloscopio mediante PC
Tensiones (V)
-0,1
0 50 100 150 200 250 300 350 400 450 500 550 600 650 700 750 800
Canal 4 Dispositivo 1 Unidades ms
Canal: 4
Número de muestras: 40
Dispositivo: 1
Medición continua
Nivel de disparo: -0,12 V
Periodo de muestreo
1 us 2 us 5 us 10 us 20 us 50 us 0,1ms 0,2ms 0,5ms 1ms 2ms 5ms 10ms 20ms 50ms 0,1s 0,2s 0,5s
Estado: Medición en proceso
Comenzar
Cerrar

Figura 7.- Aparición del efecto "aliasing".

mente, mostrando unidades de tiempo en función del número de puntos por barrido y la frecuencia de muestreo. En la parte inferior de la gráfica se indica el canal analógico sobre el que se mide, el dispositivo de adquisición de datos que está siendo utilizado (puede haber varios), y las unidades correspondientes al eje de tiempos. Como se ha podido observar en la Figura 3, la aplicación es capaz de representar con gran precisión las señales adquiridas.

CONTROL DEL NIVEL DE DISPARO

El nivel de disparo indica el valor de tensión a partir del cual comenzará cada barrido. Cuando se concluye un barrido horizontal completo, el osciloscopio permanece a la espera hasta que se alcanza la tensión de disparo, momento en el cual se inicia un nuevo barrido. El control del nivel de disparo se aloja debajo del panel gráfico, en la parte derecha de la ventana. La casilla de verificación permite habilitar o deshabilitar el disparo por nivel. Si dicha casilla no está marcada, el osciloscopio iniciará una nueva representación tras alcanzar el extremo derecho del panel gráfico, sin espera alguna.

El osciloscopio está virtualmente integrado en el sistema en desarrollo

El disparo por nivel resulta muy útil, ya que al representar señales periódicas es común que un barrido completo no muestre un número entero de periodos de la señal medida. En ese caso, el trazado comenzará cada vez con un valor de tensión diferente, dando la sensación de que la señal se desplaza por la pantalla, y por tanto dificultando la visualización de resultados (sobre todo los cálculos sobre el eje de tiempo). La solución consiste en fijar el nivel de disparo a un valor por el que la señal pase periódicamente.

En la Figura 5 se observa una señal *DC* negativa afectada por el ruido de la red eléctrica (obsérvese que el periodo de la señal es de 20 *ms*, lo que son 50Hz de frecuencia). El nivel de disparo se ha fijado a -0,112*V* y por dicho valor comienza cada barrido. La señal permanece inmóvil en pantalla, siendo posible realizar mediciones de tiempo. Si se desactivara el disparo por nivel, la señal comenzaría a moverse y sería difícil medir su periodo. Si se hubiera seleccionado un nivel de disparo de 0*V* o de -1*V* el disparo nunca se efectuaría, ya que la señal nunca alcanza dichos valores.

Dada la importancia del nivel de disparo, el usuario puede cambiar su valor para ajustarlo a la cantidad precisa, incluso durante el proceso de muestreo.

Es conveniente indicar que los osciloscopios comerciales incorporan multitud de opciones de disparo (como el disparo por flanco). Se invita al usuario a ampliar el código fuente de este programa para dar soporte a nuevos tipos de disparo no soportados en esta primera versión.

CONTROL DEL MODO DE MEDICIÓN

El osciloscopio puede trabajar en dos modos: continuo o barrido simple. En modo continuo, tras cada barrido se inicia uno nuevo (aplicando las condiciones de disparo si procede), tal y como ocurre en un osciloscopio de laboratorio usual. En modo de barrido simple, sólo se adquieren las tensiones correspondientes a un barrido y se muestran en pantalla. El modo continuo tiene utilidad para monitorizar señales periódicas. Si, en cambio, se desea capturar una señal transitoria, conviene efectuar un único barrido. El disparo por nivel se sigue aplicando en el modo de barrido simple, si se halla activado.

Por ejemplo, para cazar un pico de tensión aislado de 0 a 5 *Voltios*, se podría fijar el nivel de disparo a 0,1*V* y lanzar un barrido simple. De esta forma, el barrido se dispararía al comenzar el pico, y quedaría presentado en pantalla para su estudio.

El valor óptimo del número de muestras depende del periodo de la señal monitorizada

El modo de medición se selecciona mediante la casilla de verificación etiquetada como *medición continua*.

CONTROL DEL NÚMERO DE MUESTRAS POR BARRIDO

El número de muestras de tensión que representar en cada barrido se introduce directamente en la casilla de texto "numero de muestras". Dicho valor solo puede ser modificado antes de iniciar la adquisición de datos. En caso de medición continua, es necesario pasar a medición discontinua y esperar a que finalice la adquisición para poder modificar el número de muestras por barrido.

El valor óptimo del número de muestras que utilizar depende generalmente del periodo de la señal que se monitoriza. Un valor excesivamente reducido impedirá observar al completo la señal (solo se observaría una fracción del periodo). Por el contrario, un valor excesivamente alto puede llevarnos a observar demasiados periodos de la señal, que al representarse en pantalla produce una perdida total de información, y puede llegar a consumir un tiempo excesivo. En la práctica, conviene fijar un número de muestras que permita visualizar unos 3 periodos de la señal, con lo que se obtiene toda la información necesaria con buen rendimiento.

CONTROL DEL PERIODO DE MUESTREO

El periodo de muestreo indica el intervalo de tiempo transcurrido entre la adquisición de dos tensiones consecutivas. Hablando en otros términos, determina la precisión del eje de tiempos. Aplicando el teorema del muestreo, si deseamos visualizar una señal cuyo mínimo periodo es T, habrá que muestrear con un periodo menor que T/2 para que la señal se presente sin pérdida de información. En la práctica, para visualizar una señal de periodo T, escogeremos un periodo de muestreo T/10. Por ejemplo, una sinusoide de 1 KHz (periodo de 1ms) se verá perfectamente en pantalla si utilizamos un periodo de muestreo de 0,1 ms. Si no se respeta el teorema del muestreo, aparece el fenómeno de *aliasing* (perdida de información).

BUSCAMOS COLABORADORES

Si además de leer

SÓLO PROGRAMADORES

te gusta la programación, y quieres escribir en tu revista, no dudes en ponerte en contacto con nosotros. Envíanos tu propuesta junto a tu curriculum a la siguiente dirección:

REVISTAS PROFESIONALES
C/San Sotero, 5 - 1.ª planta
28037 Madrid

Ref.: Colaboraciones Sólo Programadores

E-mail: solop@virtualsw.es

Para contemplar dicho efecto con un ejemplo, véase la Figura 5, en la que aparece una señal DC negativa con los 50Hz de la red eléctrica superpuestos y algo de ruido de alta frecuencia. Si la máxima frecuencia que deseamos ver con precisión es 50Hz (periodo de 20 ms), deberíamos escoger un periodo de muestreo de 2 ms para ver correctamente la señal. En la Figura 5 se aprecia la selección de un periodo de muestreo de 50 microsegundos (20 KHz), lo que permite ver con precisión señales de hasta 2 KHz de frecuencia. Por esto se aprecia un ruido superpuesto de frecuencias mayores que los 50Hz de la red eléctrica.

Disminuyamos ahora la frecuencia de muestreo hasta 2KHz (periodo de muestreo: 0,5 ms). Esto permitirá contemplar señales de hasta 200Hz. El resultado se muestra en la Figura 6. Como se puede observar, ahora el pequeño ruido de alta frecuencia ya no es visible, puesto que queda por encima de los 200 Hz que podemos ver como máximo y es de bajo nivel de tensión. Si el ruido fuera de mayor nivel, aparecería superpuesto como si se tratara de una señal de baja frecuencia (el efecto de aliasing se dejaría ver, haciéndonos perder información).

El osciloscopio trabaja en modos continuo o barrido

Si llegamos al caso extremo de dejar la frecuencia de muestreo en 50Hz, habremos llegado al límite impuesto por el teorema de muestreo, perdiendo información Esto se puede observar en la Figura 7, donde la señal se presenta como si fuera un nivel DC negativo (cuando realmente no lo es).

Aún se puede ir mas allá, aumentando el periodo de muestreo y observando cómo el osciloscopio nos "engaña" cada vez más.

Queda demostrado que es fundamental escoger un periodo de muestreo suficientemente pequeño si se desea obtener resultados fiables. También es muy importante hacer una buena elección del número de muestras para poder observar la señal con claridad y con el menor retardo posible entre barridos.

INDICADOR DE ESTADO

En la parte inferior de la ventana se encuentra un cuadro de texto que informa sobre el estado del proceso. A continuación se listan algunos de los aspectos importantes sobre los que informa la aplicación:

- Durante la adquisición de datos, se muestran los dos posibles estados: esperando al disparo del barrido horizontal o medición en proceso. Si la aplicación se detiene demasiado tiempo en espera del disparo, es muy probable que el nivel de disparo seleccionado no se pueda alcanzar. La solución consiste en reajustar el nivel de disparo, o desactivar el disparo por nivel.

- Los errores graves de adquisición de datos lanzados por la librería *NI-DAQ* se muestran a través de su código de error. En la ayuda de *NI-DAQ* existe una sección donde se listan los códigos de error y su significado.

- El programa también informa en caso de ocurrir errores leves. Dichos errores dejan continuar con la operación en curso, pero esta puede no concluir satisfactoriamente.

LICENCIA DE USO

La aplicación *Osciloscopio.exe* se presenta acompañada del correspondiente código fuente. El usuario es libre de utilizar el programa en las aplicaciones que desee, examinar el código fuente, y modificarlo para mejorarlo o adaptarlo a sus necesidades, siempre que se respeten las condiciones detalladas en la licencia.

Es importante leer las indicaciones del fabricante para no dañar la tarjeta u otros circuitos.

CONCLUSIONES

Nos hemos introducido en el mundo de la instrumentación virtual. Hemos comenzado por las capas inferiores, conociendo como funciona una tarjeta de adquisición de datos.

Después, hemos subido al siguiente nivel, con una librería de adquisición de datos de reconocido prestigio.

El paso siguiente nos ha llevado a un nivel de abstracción mayor, desarrollando un componente *VCL* que hace muy simple y rápida la creación de instrumentos virtuales: oculta totalmente la complejidad de la adquisición de datos, y nos permite centrarnos en otros aspectos del desarrollo.

Finalmente, se han mostrado dos ejemplos de aplicación realmente útiles (situándonos así en la capa superior del modelo). El lector puede tomar los conocimientos adquiridos y el componente desarrollado como base para crear nuevos instrumentos virtuales que resuelvan sus necesidades.

De rienda suelta a su creatividad e imaginación con:

MASTER COREL DRAW

Suscríbase ahora al Master Corel Draw, y junto a la colección de 50 fascículos y 50 CD-ROM recibirá la *Suite completa de Corel Draw 9.0, versión estudiante.*

Más de 1.000 páginas de teoría y práctica llenas de los más útiles consejos para convertirse en un auténtico Master del diseño gráfico.

Más de 200 programas y utilidades del diseño gráfico, más de 500 fotografías profesionales y decenas de Webs de diseño para navegación off-line.

Ejercicios basados en las explicaciones del curso que le ayudarán a perfeccionar su técnica.

Diploma acreditativo de haber recibido y aprovechado el *Master Corel Draw* de diseño gráfico.

Con 5 carpetas para almacenar sus fascículos y un archivador para ordenar los CD-ROM.

Desarrolle toda la capacidad de las nuevas herramientas que le ofrece Corel Draw 9: Ilustraciones vectoriales, Corel Photo-Paint 9 (edición de imágenes y bitmaps), C. Texture (Texturas realistas y naturales), C. Capture (Aplicación para la captura de pantallas), 25.000 imágenes clip-art de alta calidad (10.000 nuevas para C. Draw 9), y muchas otros extras.

RECORTE Y ENVÍE EL CUPÓN ADJUNTO A:

Revistas Profesionales, S. L. • San Sotero, 5-1.ª planta • 28037 • Madrid • Fax: 91 327 13 03

BOLETÍN DE SUSCRIPCIÓN

OFERTA VÁLIDA SÓLO PARA ESPAÑA

SI, deseo suscribirme al **MASTER COREL DRAW**

CONDICIONES DE PAGO (MARQUE LA OPCIÓN DESEADA)

- ❑ 50 fascículos + 46 CD-ROM 795 Ptas/u. **TOTAL: 39.750 PTAS.**
- ❑ 5 carpetas para archivar fascículos a 950 Ptas./u. **TOTAL: 4.750 PTAS.**
- ❑ 1 carpeta para archivar CD´s a 1.250 Ptas.

VALORACIÓN TOTAL DE LA COLECCIÓN CON COREL DRAW 9 INCLUIDO: 84.750 PTAS.

- ❑ Pago al contado. Ahorro del **55%. TOTAL: 38.140 Ptas.**
- ❑ Pago en 3 plazos. Ahorro del **45%**. Cada plazo: 15.535 Ptas. TOTAL: 46.610

ENTREGA DEL COREL DRAW 9 EN EL ÚLTIMO ENVÍO

Sin Corel Draw 9 valoración total: 45.750 Ptas.

- ❑ Pago en 6 plazos (con ahorro del **25%**): Cada plazo: 5.720 Ptas. TOTAL: 34.315
- ❑ Pago en 12 plazos (con ahorro del **20%**): Cada plazo: 3.050 Ptas. TOTAL: 36.600

FORMAS DE PAGO:

❑ Con **cargo a mi tarjeta VISA** Nº ____/____ /____ /____ Caduca .....

❑ **Domiciliación bancaria**

Ruego a Ud. que se sirva cargar en mi ❑ cuenta corriente ❑ libreta de ahorro el/los recibos que le serán presentados por REVISTAS PROFESIONALES, S.L. como pago de mi suscripción al **MASTER COREL DRAW DE DISEÑO GRÁFICO**

CÓDIGO CUENTA CORRIENTE

ENTIDAD	OFICINA	DC	Nº CUENTA

❑ **Cheque** a nombre de REVISTAS PROFESIONALES, SL.

❑ **Contra reembolso** del importe más gastos de envío.

❑ **Giro Postal** (adjunto fotocopia del resguardo)*.

Firma

*Esta forma de pago es exclusivamente para pagos al contado

UTILICE MAYÚSCULAS PARA RELLENAR ESTA TARJETA

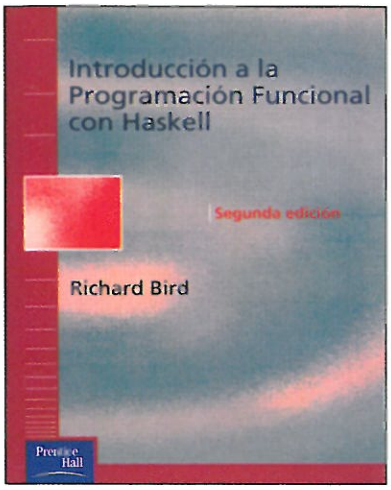
Nombre y apellidos .....F. nacimiento.....

Domicilio ..... C.P. .....

Ciudad ..... Provincia .....Telf. .....

e-mail ..... Profesión .....

PROGRAMACIÓN EN WINDOWS 2000



Segunda edición, revisada y actualizada, del excelente trabajo de *Richard Bird*, en la que se establecen los fundamentos de la programación funcional, a través del lenguaje *Hakell*, uno de los de mayor aceptación en la presentación de algoritmos funcionales.

El libro está dirigido especialmente a estudiantes de primer año, ya que en él se aborda en primer lugar la programación en general, sin ser necesarios conocimientos previos de programación o informática. No obstante, los programadores con cierta experiencia encontrarán en esta obra una perfecta referencia a las bases de la programación funcional.

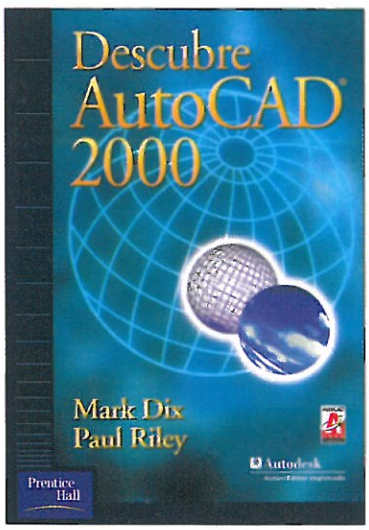
Esta actualización, además de estar reorganizada, incluye apartados nuevos dedicados a la optimación de programas, programación monádica, y tipos abstractos de datos, con lo que nos encontramos ante una obra más completa que la anterior.

La traducción del profesor *Peña Marí* es excelente, con oportunas notas a pie de página que adecuan perfectamente las expresiones inglesas a nuestra lengua.

Editorial: Prentice Hall
Nº de páginas: 369
Nivel: Medio

Autor: Richar Bird
Idioma: Español
Precio: 3.800Ptas. (I.V.A inc.)

DESCUBRE AUTOCAD 2000



Completo manual destinado a introducir a los diseñadores en el manejo y dominio de *AutoCad*. El temario se desarrolla sobre una estructura de dificultad creciente, permitiendo al lector asimilar los conocimientos adquiridos mediante la experiencia y la práctica a través de ejercicios.

Si bien los ejercicios están en inglés, van perfectamente explicados, para que no encontremos grandes dificultades en la comprensión de comandos y técnicas. La orientación claramente pedagógica de la obra, se ve incrementada por una sección de preguntas y autoevaluación.

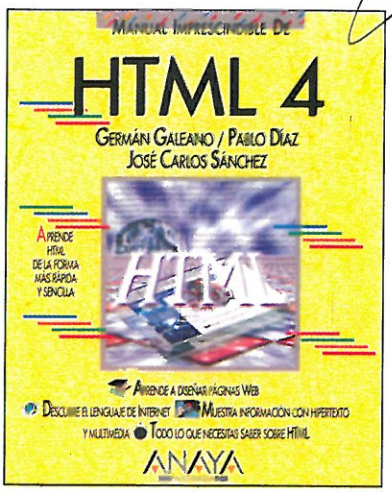
Por otra parte, el acceso a *Internet* nos permitirá entrar en la *Web* de consulta, donde encontraremos nuevos proyectos relacionados con cada capítulo del libro.

En este libro encontraremos, además, multitud de consejos, trucos y atajos que permitirán aprovechar todas las posibilidades de esta potente herramienta de diseño.

Editorial: Prentice Hall
Nº de páginas: 761
Nivel: Medio y Avanzado

Autor: Mark Dix y Paul Riley
Idioma: Español
Precio: 4.475 Ptas. (I.V.A inc.)

HTML 4


MANUAL IMPRESCINDIBLE DE
HTML 4
GERMÁN GALEANO / PABLO DÍAZ
JOSÉ CARLOS SÁNCHEZ
APRENDE HTML DE LA FORMA MÁS RÁPIDA Y SENCILLA
APRENDE A DISEÑAR PÁGINAS WEB
DESCUBRE EL LENGUAJE DE INTERNET
MUESTRA INFORMACIÓN CON HIPERTEXTO Y MULTIMEDIA
TODO LO QUE NECESITAS SABER SOBRE HTML
ANAYA

Guía práctica para diseño, optimación y homologación de páginas *Web* a través del Lenguaje *HTML*. Estas páginas son un excelente medio para adentrar al profano en el proceso de representación multimedia e *Internet*, además de servir a los más experimentados para refrescar sus conocimientos.

Este manual, ilustrado con útiles ejemplos, empieza por estudiar las nociones más elementales, para terminar desarrollando las características avanzadas que nos permitan realizar nuestros diseños de la manera más funcional y adecuada a nuestras necesidades.

Enlaces, mapas, tablas y formularios; *Java, ASP y XML...*, en el manual de *HTML 4* de Anaya está todo lo que debes saber para estar al día en el diseño *Web* y la información con hipertexto y multimedia.

Para dominar *HTML* en su versión *4.01*, sólo necesitaremos este manual, un *PC* y un visor como *Navigator* o *Explorer*.

Editorial: Anaya Multimedia
Número de páginas: 416
Nivel: Medio

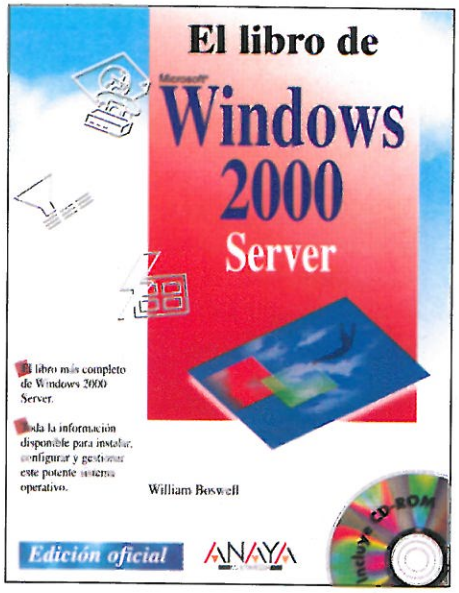
Autor: G. Galeano, P. Díaz y J. C. Sánchez
Idioma: Español
Precio: 3.800 Ptas. (I.V.A. inc.)

EL LIBRO DE WINDOWS 2000 SERVER

Este libro pretende ser la primera publicación de referencia para la administración profesional de *Windows*, para lo que se ha tenido especial cuidado en describir los procedimientos y prácticas más eficientes de instalación, configuración, despliegue, y gestión de servidores y escritorios de *Windows 2000*.

El autor presenta cada capítulo con un peculiar estilo que hace más amena su lectura y comprensión. De esta forma, empezaremos por la instalación y configuración del sistema, para llegar a la administración de recursos compartidos y de entornos de usuario, pasando por la seguridad de acceso a redes y *Kerberos*, y los servicios del *Directorio Activo*.

El libro viene acompañado de un *CD-Rom* con *freeware, shareware*, y copias de evaluación de programas, que nos ayudarán a descubrir todas las posibilidades de *Windows 2000 Server*.


El libro de
Windows 2000 Server
El libro más completo de Windows 2000 Server.
Toda la información disponible para instalar, configurar y gestionar este potente sistema operativo.
William Boswell
Edición oficial
ANAYA
Incluye CD-ROM

Editorial: Anaya Multimedia
Nº de páginas: 832
Nivel: Medio

Autores: William Boswell
Idioma: Español
Precio: 5.850 Ptas. (I.V.A. inc.)

Contenido del CD-ROM

ENTORNOS DE PROGRAMACIÓN

3D GAMESTUDIO 4.1
Sistema avanzado de desarrollo de aplicaciones *2D* y *3D*. Ofrece a los no programadores la posibilidad de crear juegos de diferentes tipos.

BORLAND C++ COMPILER 5.5
Rápido compilador de *C++* en 32 *bits*. Incluye la última versión para *ANSI/ISO C++*, y plantillas *RTL* y *framework STL*.

DOC-TO-HELP 2000 INTERNATIONAL
Convierte *Microsoft Word* en un potente entorno de autor para ficheros de ayuda de *Windows*.

EDITORPRO SUITE 1.5
Completo editor de texto que soporta nueve lenguajes. Edita ficheros de texto, *Delphi, Paradox, C, C++, HTML, Java, SQL* y *Visual Basic*.

INPRISE APPLICATION SERVER 4.0
Colección de servicios y herramientas para administración de *Webs*, con todas las funciones necesarias para publicar y gestionar *Webs*.

INTERBASE 5.6
Base de datos *SQL* de alto rendimiento, para trabajar con aplicaciones de negocios, equipos móviles y programas de *Internet* en entornos *Linux, Windows, Solaris y UNIX*.

LENGUAJES

VISUAL BASIC

COYOTE SOFTWARE WIZARD 2000 V1.1
Colección de herramientas que agiliza la creación e inclusión de asistentes en nuestras aplicaciones.

FAST.LIB 2000
Librería de controles *ActiveX* y clases *OLE* para la programación en *Windows*. Incluye más de 100 controles de usuario y 20 clases para *Visual Basic*.

LOCKNUTS 1.0
Programa de criptografía para programadores de *Visual Basic*. Codifica y guarda de forma protegida nuestra información.

JAVA

COCKTAIL STANDARD EDITION 1.2
Utilidad para crear aplicaciones *Java*, muy sencilla, ya que se trata de añadir nuevos componentes a una estructura en forma de árbol.

JFCSUITE 3.5
Colección de componentes *JavaBeans*. Añaden nuevas funciones al estándar *Java Foundation Classes*, como una rejilla de datos jerárquicos y una librería de extensiones de datos de entrada.

RMAIL 1.0
Librería *Java* para enviar y recibir correo electrónico con formato *MIME*. Soporta *SMTP, POP3* y *MIME v1.0*.

HTML

EXCELON STYLUS 2.0
Editor visual para hojas de estilo *XSL (eXtensible Stylesheet Language)*.

SCRIPTWORX 4.1
Completo editor *HTML* que incorpora un servidor *Web* y un cliente *FTP*.

XML SPY 3.0
Potente editor *XML* y *DTD*. Proporciona cuatro vistas avanzadas de los documentos, una ventana de proyectos, validación *DTD*, opciones de impresión y de búsqueda y sustitución.

OTROS

IMAGE++ 2.1
Herramienta de desarrollo orientado a objetos. Muy adecuada para procesar imágenes digitales.

IOCOMP PROFESSIONAL INSTRUMENT ACTIVEX CONTROLS 1.1.2
Colección de componentes para aplicaciones de instrumentación.

NCTWAVRECORD 1.02
Graba y reproduce ficheros *WAV* y *MP3*. Este control *ActiveX* graba ficheros *WAV* desde un micrófono en cualquier formato.

OXYGEN PLAN LIBRARY 1.2
Programa para construir planos interactivos. Cuenta con una gran cantidad de objetos totalmente personalizables.

HERRAMIENTAS Y UTILIDADES

AUTOMATE PRO 4.4
Programador de aplicaciones, programa de avisos y lenguaje de macro.

BCX 1.7
Conversor de *Basic* a *C*. *BCX* permite editar programas en *BASIC* y convertir, posteriormente, su código fuente en *ANSI C*.

DATA MANAGER 2.5
Utilidad de procesamiento de datos. Presenta una interfaz gráfica muy intuitiva que facilita la manipulación de los datos.

DLL EXPLORER 2.2.2
Genera listas de las librerías y otros módulos que el sistema ope-

rativo está utilizando. Muestra el estado actual del sistema operativo.

HYPERTEXT COMPILER 1.0
Compila documentos *HTML* en una única página *HTML*, que emula un sitio *Web* y funciona como un índice, ejecutando las páginas individuales.

MOUNTFOCUS KEYBOARD DESIGNER 1.0
Utilidad para crear teclados virtuales, que pueden ser usados con cualquier aplicación *Windows* que trabaje con pantallas táctiles o en un entorno normal.

PICTURETAKER ENTERPRISE EDITION 3.01
Crea copias de seguridad de los ficheros críticos del sistema.

TRACEPLUS32 WEB DETECTIVE 1.52.002
Herramienta de análisis y seguimiento para el desarrollo de sitios *Web*. Decodifica el protocolo *HTTP*.

REDES

MDAEMON 3.0.4.0
Servidor *SMTP* y *POP3* que incorpora un *firewall* de correo electrónico.

MONITORIT 2.0
Controla el rendimiento de sistemas, servidores *Windows* de 32 *bits* y servicios *Web*.

NETWORK UNPLUGGED 2.0B
Herramienta para usuarios de ordenadores portátiles. Copia entre el portátil y el equipo de sobremesa, los ficheros de red.

PROBOT 1.00
Utilidad de monitorización del sistema. Es útil para su uso en intranets e *Internet*.

PROXY+ 2.40
Servidor *proxy*, de correo y *firewall*. Protege la red de ataques e intentos de acceso no autorizados, trabaja con casi todos los tipos de conexiones a Internet

SNIFFER 2.9
Intercepta y analiza los paquetes transmitidos en una red. El programa se instala en cualquier ordenador de la red local.

WINGATE 4.0.1
Conecta a los usuarios de una red local a *Internet*. Con una misma conexión, los usuarios pueden disponer de servicios de correo electrónico.

DOCUMENTACIÓN/ TUTORIALES

10 MINUTE GUIDE TO PRACTICAL UNIX 9.3
Tutorial de *UNIX*. Es muy adecuado para todos aquellos que necesiten trabajar en diferentes sistemas *Unix* cada día.

HTML TUTORIAL 1.1
Tutorial de *HTML* para principiantes. Ofrece numerosos ejemplos y describe cada elemento *HTML* utilizado.

ITTUTOR 2000 V3.0
Herramienta de aprendizaje para las certificaciones *MCSE* y *MCSD*. Incluye información y pruebas para materias relativas a certificaciones *MCSE* y *MCSD*.

MAZEBOTS 1.21
Programa de aprendizaje de robótica. Ayuda a aprender habilidades de programación y a desarrollar aplicaciones de inteligencia artificial.

TEACH ME HTML 1.0
Tutorial de HTML. Presenta una interfaz sencilla y amigable que facilita el aprendizaje de los principios de la programación en *HTML*.

IMPRESCINDIBLES

ANTIVIRUS
AntiViral Toolkit Pro 3.0.129
McAfee VirusScan 5.0.1
Panda Antivirus 6.18.00 Platinum
Quick Heal 5.24b

GRÁFICOS
Icon Bank 4.0 Gold Edition Web Edition / Desktop Edition
IconForge 4.7
MicroAngelo 98 v4.77
Paint Shop Pro with Animation Shop 6.02
Reptile 2.01
SureThing CD Labeler 2.10.15
ThumbsPlus 4.10-S
Ultra Fractal 2.04
Xara WebStyle 1.2
Xara 3D 4.0

INTERNET
Copernic 2000 v4.55
Cuentapasos 3.79
CuteFTP 4.0
Dial-Up Magic 1.8
Eudora Pro 4.3.2
Free Agent 1.21
GetRight 4.2b
Go!Zilla 3.5
Guardián 1.1
HotDog Professional 6.1
ICQ 2000a Beta v.4.31 build #3143
i.Share 3.5
Jet Color 1.20
Mass Downloader 1.2.95-SR2
MIRC 32 5.71
SubmitWolf ES 4.02.002)
WebZIP 3.71
XMLwriter 1.21

MULTIMEDIA
AudioCatalyst 2.01
COWON Jet-Audio 4.7
MusicMatch Jukebox 5.1
Sonique 1.51
WinAMP 2.64

NAVEGADORES
Internet Explorer 5.01
Netscape Communicator 4.72
Opera 3.62

PROGRAMACIÓN
Decafe Pro 3.6
Hackman 4.04
Help & Manual 2.52
Java Development Kit 1.2.2-001
UltraEdit Professional Text/HEX Editor 7.10
Windows Registry Guide 1.3
WinHex 9.45

UTILIDADES
Adobe AcrobatReader 4.05
DirectX 7.0a
Emergency Recovery System 9.05
Nero 5.0.0.9
Windows Commander 4.03
WinZip 8.0

Dudas técnicas

En esta sección, como cada mes, SÓLO PROGRAMADORES os brinda la oportunidad de encontrar respuesta a las dudas que podáis tener en algún asunto relacionado con la programación en cualquier entorno de desarrollo.

PREGUNTA

Hola amigos de *Sólo Programadores*:

Estoy haciendo mis páginas *Web* y me ha surgido una duda con la etiqueta <SELECT> de los formularios. Me gustaría saber si es posible añadir y eliminar dinámicamente elementos de las listas que se hacen con esa etiqueta. También querría que apareciese seleccionado por defecto un elemento en función del valor de una variable. Por último, quería aplicar estilos a las listas desplegables, aunque no estoy muy seguro de que esto pueda hacerse.

Muchas gracias.

Alfonso Ruiz.

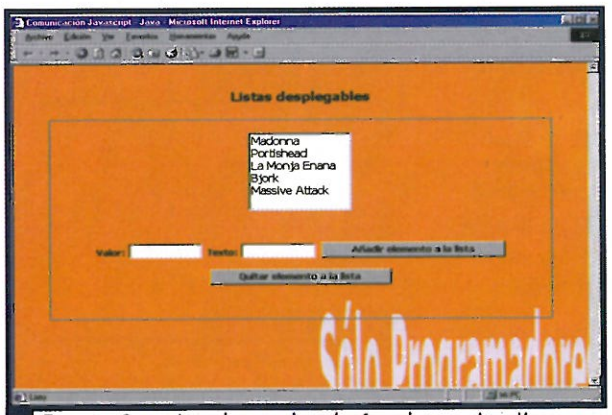


Figura 1. - Se pueden aplicar estilos a los formularios, si bien *Netscape Communicator 4.x* sólo interpreta los relativos al tipo de letra.

RESPUESTA

Estimado lector:

Efectivamente es posible hacer lo que tu dices, si bien hay algunas diferencias notables en función del navegador utilizado, *Netscape Communicator* 4.x o *Internet Explorer* 5.x. Empezaremos mostrando cómo se pueden añadir elementos a la lista desplegable. Supongamos que tenemos la lista descrita por el siguiente código *HTML*:

```
<SELECT NAME="LISTADESPLEGABLE"
        SIZE="6">
<OPTION VALUE="45361">Madonna
...
</SELECT>
```

Todos los elementos de una lista desplegable están formados por dos componentes: el valor asociado correspondiente a ese elemento, y la cadena de texto que se muestra al usuario. Para añadir un elementos a una lista primero hay que crear dicho elemento:

```
var o = new Option("La Monja
    Enana","76813");
```

El paso siguiente consiste en añadir el elemento a la lista. Esto puede hacerse:

```
f.LISTADESPLEGABLE.options[f.LIS-
    TADESPLEGABLE.options.length]
    = o;
```

Suponiendo que *f* es una variable que contiene la referencia correspondiente al formulario donde se ubica la lista desplegable. Nótese que los *arrays* en *Javascript* van desde *0* hasta *n-1*, donde *n* es el tamaño del mismo, y por lo tanto la posición adecuada donde insertar un elemento.

Existe un problema con lo anterior, cuando añadimos a la lista un elemento con una cadena de texto demasiado larga, *Internet Explorer* adaptará dinámicamente el tamaño del componente dentro de la página, mientras que *Netscape Communicator,* no.

Para eliminar un elemento de la lista lo único que debemos hacer es asignar el valor *null* al elemento correspondiente, tal y como se muestra en el código siguiente:

```
f.LISTADESPLEGABLE.options[i] =
    null;
```

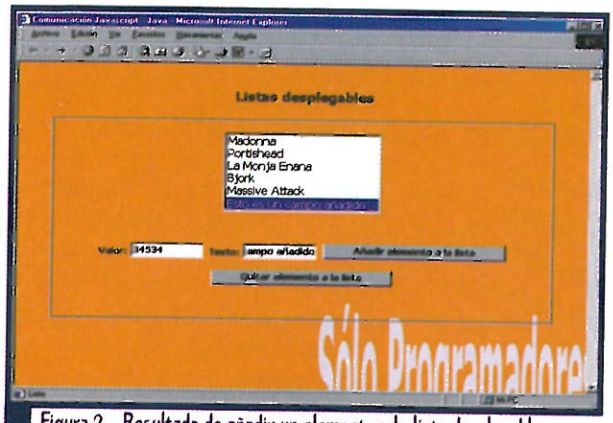


Figura 2. - Resultado de añadir un elemento a la lista desplegable.

Donde *i* corresponde a la posición del elemento que se desea borrar de la lista desplegable.

Para seleccionar un elemento por defecto, en función de una variable de *Javascript* puedes hacerlo mediante un procedimiento que se ejecute cuando la página se termine de cargar:

```
<BODY onload="Configurar()">
```

Ese procedimiento podría ser como sigue:

```
function Configurar() {
   var i;
   var f =
    document.forms["MIFORM"];
   var l =
    i<f.LISTADESPLEGABLE.options.
    length;
   for (i=0;i<l;i++) {
     if
    (f.LISTADESPLEGABLE.options[i
    ].value == mivar) {
       f.LISTADESPLEGABLE.selec-
    tedIndex = i;
     }
   }
}
```

Por último, es posible aplicar estilos a las listas desplegables y en general a cualquier elemento de un formulario. Ahora bien, como tú mismo apuntas, existen problemas, especialmente con *Netscape Communicator* 4.x que todavía no soporta el estándar *CSS* adecuadamente. A grandes rasgos se pueden decir que los únicos atributos que no presentan demasiadas dificultades son los relativos a la fuente. Para aplicar un estilo, simplemente tienes que crearlo en la cabecera de la página. Un ejemplo podría ser:

```
<STYLE TYPE="text/css">
SELECT  {
  font-family : Tahoma;
  font-size : 12pt;
}
</STYLE>
```

PREGUNTA

Hola *Sólo Programadores*:

Hace poco tiempo llegó a mis manos un número de vuestra revista que es un poco antiguo. En él había un artículo sobre *WSH* (Windows Scripting Host). Supongo que no termino de entender algunas cosas porque no he podido leer todos los artículos de la serie, pero según lo que yo entendido es que se pueden hacer programas con *Javascript*, y que esos programas pueden hacer llamar a otras aplicaciones, "hablar" con *Windows*, etc. ¿Es esto así? ¿Qué necesito para empezar a hacer programas *WSH*? ¿Dónde puedo encontrar información?

Laura Ascensión.

RESPUESTA

Estimada lectora:

La tecnología *WSH* se puede resumir a grandes rasgos de la siguiente manera: dota a los sistemas *Windows* de un lenguaje de *script* (puede ser *JScript* o *VBScript*) que proporciona una manera de hacer lo que antes denominaban ficheros *BAT*. Lo anterior no es más que un resumen, ya que en realidad se pueden llevar a cabo otras muchas tareas.

Por ejemplo, supongamos que tenemos el siguiente código:

```
var ie = new
    ActiveXObject("InternetExplo-
    rer.Application");

ie.left=50;
ie.top=50;
ie.height=300;
ie.width=400;
ie.toolbar=false;
ie.StatusBar=false;

ie.navigate("C:\\MIPAGI1.HTM");
ie.visible=1;
```

Y lo guardamos en un fichero de texto con el nombre *wsh_ejemplo.js*. Cuando hagamos doble clic en ese fichero, se abrirá una ventana de *Internet Explorer*, situada en la posición (50,50) de nuestra pantalla, con una achura de 400 píxeles y una altura de 300 píxeles, sin la barra de herramientas ni la barra de estado. En esa ventana además se cargará la página *MIPAGI1.HTM*, que se encuentra en la raíz del disco duro etiquetado como *C*.

El ejemplo anterior es solamente una muestra del gran número de tareas que se pueden llevar a cabo con *scripts* de *WSH*: trabajar con ficheros, directorios, unidades de red, etc. También se puede ejecutar aplicaciones externas, interaccionar con las aplicaciones de *Microsoft Office*, automatizar tareas, etc.

Como habrás podido observar, para programar *scripts* de *WSH*

solamente necesitas un editor de texto simple y corriente. *WSH* está incorporado a los sistemas *Windows* a partir de *Windows* 98, y puede instalarse en cualquier caso descargando la aplicación correspondiente del sitio *Web* de *Microsoft*.

Direcciones interesantes de *WSH* son las siguientes:

Microsoft Scripting Technologies
msdn.microsoft.com/scripting/

Win32 Scripting
cwashington.netreach.net

Windows Scripting Host FAQ
wsh.glazier.co.nz/frame.htm

En ellas podrás encontrar la documentación necesaria así como multitud de ejemplos.

PREGUNTA

Hola gente de *Sólo Programadores*:

Estoy haciendo una aplicación para gestionar los pedidos que recibo diariamente. Tengo una base de datos con varias tablas. Una de ellas se llama PEDIDO y tiene los siguientes campos: CODIGOPEDIDO, FECHAPEDIDO, CODIGOEMPRESA, CODIGOPRODUCTO y CANTIDAD. A esta base de datos accedo mediante sentencias *SQL*. No tengo demasiada experiencia y cuando voy a realizar algunas consultas en particular no sé muy bien cómo hacerlas. Concretamente:

- Quiero sacar una lista con todos los código de empresas y el número de pedidos en total que me han hecho. Tengo otra tabla de la base de datos que se llama EMPRESA y que tiene entre otros los campos CODIGOEMPRESA y NOMBREEMPRESA. ¿Cómo puedo sacar la lista anterior pero en vez de con los nombres con los códigos de empresa?
- También necesito generar una lista con todos los productos que en total han sido demandados en una cantidad mayor a un número dado.

Manuel Peñarón.

RESPUESTA

Estimado lector:

Las consultas *SQL* pueden llegar a complicarse considerablemente así que no es extraño que tengas algunos problemas con consultas algo complejas especialmente si son tus primeros pasos.

Para resolver la primera de tus consultas vamos a utilizar la función *COUNT* junto con la cláusula *GROUP BY* tal y como sigue:

```
SELECT CODIGOEMPRESA, COUNT(*) AS
    CUANTOS
FROM PEDIDO
GROUP BY CODIGOEMPRESA
```

Al aplicar la cláusula *GROUP BY* se agruparán entre sí todos los elementos de esa tabla que tengan el mismo CODIGOEMPRESA. La función *COUNT* contará precisamente el número de veces que se repite cada uno de los códigos.

En el caso de que desees obtener además el nombre de la empresa, la consulta es similar a la descrita anteriormente, con la salvedad de que se debe incluir la tabla EMPRESA.

```
SELECT NOMBREEMPRESA, CUANTOS
FROM EMPRESA, (SELECT CODIGOEM-
    PRESA, COUNT(*) AS CUANTOS
                FROM PEDIDO
                GROUP BY CODIGO-
    EMPRESA) A
WHERE EMPRESA.CODIGOEMPRESA =
    A.CODIGOEMPRESA AND
      CUANTOS > 20
```

La Tabla denominada *A* está constituida por los resultados devueltos por la primera de las consultas. Para obtener la lista deseada se hace una correspondencia entre ésta y la tabla EMPRESA, haciendo coincidir el campo CODIGOEMPRESA.

Finalmente para solucionar la última de las consultas que nos propones, hay que utilizar de nuevo la cláusula *GROUP BY*, pero esta vez con la función SUM, que realiza una suma del campo CANTIDAD para los campos CODIGOEMPRESA que coincidan.

```
SELECT CODIGOPRODUCTO, SUM(CANTI-
    DAD)
FROM PEDIDO
GROUP BY CODIGOPRODUCTO
```

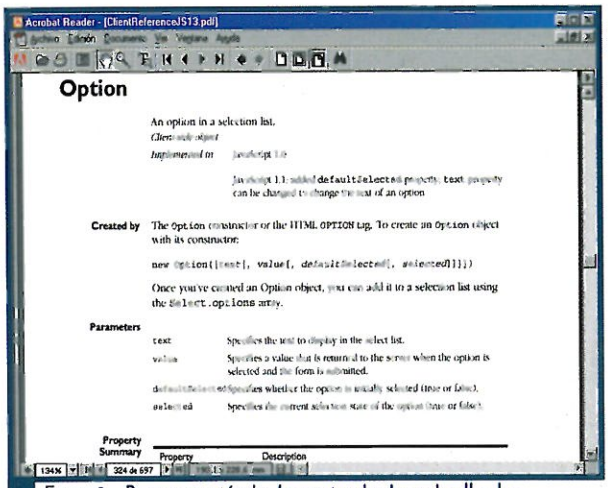
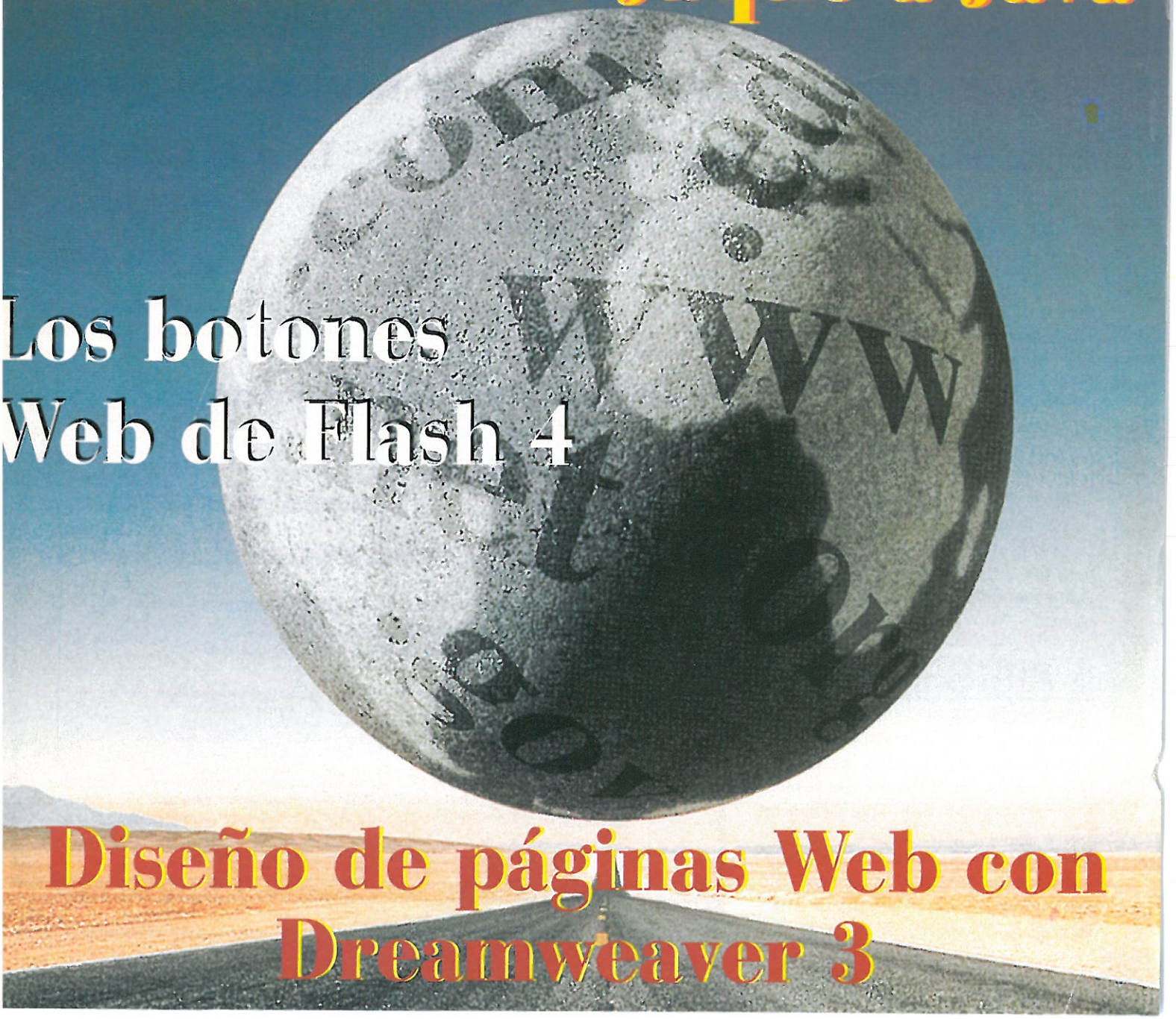


Figura 3. - Documentación de *Javascript*, donde se detallan las características del objeto *Option*.

SÓLO PROGRAMADORES INTERNET

NÚM. 1 • ESTE EJEMPLAR SE VENDE CONJUNTAMENTE CON LA REVISTA SÓLO PROGRAMADORES • Edita: Revistas Profesionales, S. L.

Microsoft C#: Jaque a Java



Los botones Web de Flash 4

Diseño de páginas Web con Dreamweaver 3

Microsoft C Sharp: Jaque a Java

Javier Sanz Alamillo.

Microsoft ha presentado recientemente *C#*, su última especificación de lenguaje. *C#* tiene como objetivo recuperar parte del mercado que acapara en la actualidad *Java*, el lenguaje de *Sun*. Pero, ¿está a tiempo de recuperar el terreno perdido? En cualquier caso, en esta competencia hay un claro ganador: el desarrollador

En la construcción de nuevos lenguajes de programación, *Microsoft* siempre ha seguido una política de adaptación a las necesidades de construcción de aplicaciones en su propio entorno. *Visual Basic (VB)* y *Visual C++ (VC++)* son los grandes abanderados en el entorno *Windows*, pero el paso del tiempo ha devaluado esta unión.

La respuesta a Java se ha hecho esperar mucho tiempo (quizá demasiado), pero por fin, los incondicionales del gigante norteamericano están de enhorabuena.

ANTECEDENTES

Al principio de la "revolución" *Internet*, el arte de la *orientación a objetos* estaba bastante avanzado, pero eran muchos los desarrolladores que se planteaban construir nuevas aplicaciones en *Visual Basic*, que al final eran origen de nuevos problemas. Y *Microsoft* no anunciaba ningún lenguaje "moderno" que se adaptara a los requerimientos del momento. Se sabía que su grupo de compiladores trabajaba con nuevos lenguajes que a la postre, por diferentes circunstancias, no acabaron de cuajar. Esto animó a compañías como *Borland* y *Sun* a lanzarse a la toma del hueco dejado por *Microsoft*.

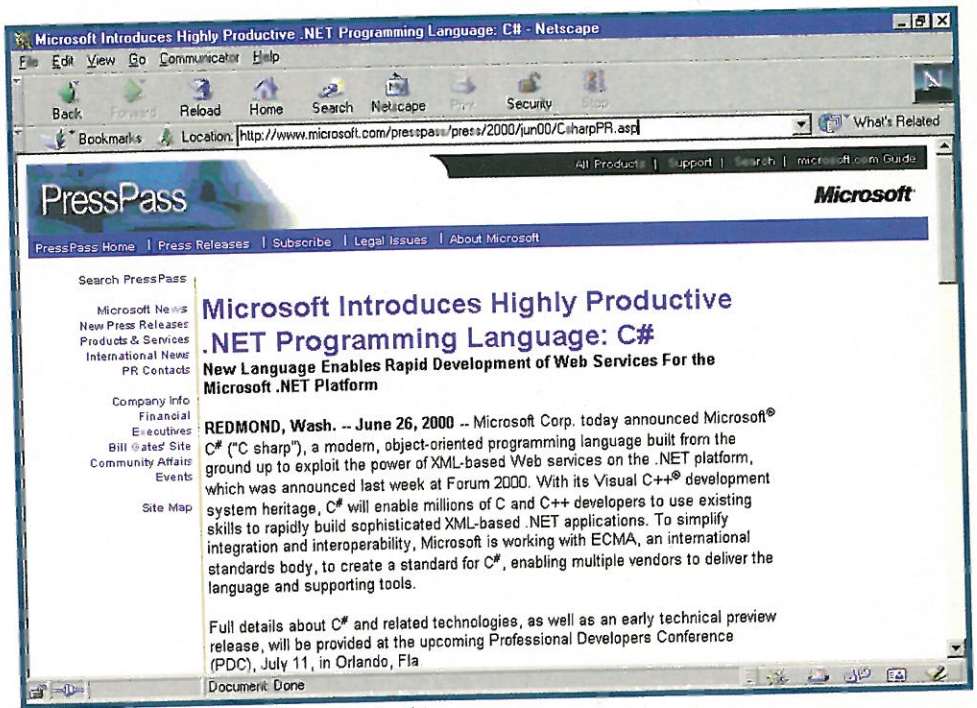

Microsoft Introduces Highly Productive .NET Programming Language: C# - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://www.microsoft.com/presspass/press/2000/jun00/CsharpPR.asp
What's Related
All Products | Support | Search | microsoft.com Guide
PressPass
Microsoft
PressPass Home | Press Releases | Subscribe | Legal Issues | About Microsoft
Search PressPass
Microsoft News
New Press Releases
Products & Services
International News
PR Contacts
Company Info
Financial
Executives
Bill Gates' Site
Community Affairs
Events
Site Map
Microsoft Introduces Highly Productive .NET Programming Language: C#
New Language Enables Rapid Development of Web Services For the Microsoft .NET Platform
REDMOND, Wash. -- June 26, 2000 -- Microsoft Corp. today announced Microsoft® C# ("C sharp"), a modern, object-oriented programming language built from the ground up to exploit the power of XML-based Web services on the .NET platform, which was announced last week at Forum 2000. With its Visual C++® development system heritage, C# will enable millions of C and C++ developers to use existing skills to rapidly build sophisticated XML-based .NET applications. To simplify integration and interoperability, Microsoft is working with ECMA, an international standards body, to create a standard for C#, enabling multiple vendors to deliver the language and supporting tools.
Full details about C# and related technologies, as well as an early technical preview release, will be provided at the upcoming Professional Developers Conference (PDC), July 11, in Orlando, Fla
Document: Done

Figura 1. - Nota prensa de Microsoft sobre C#.

En poco tiempo asistimos a la aparición del *Pascal* orientado a objetos, *Delphi,* a versiones mejoradas y completísimas de *Borland C++* que desbancaban totalmente a *Visual C++* en la mayoría de los desarrollos... Eran cada vez más los desarrolladores que dejaban *Microsoft* para pasarse a *Borland*.

C# se presenta como un rival de altura para Java

Pero todo saltó por los aires con la aparición de *Java*. *Sun* presento un día una especificación sobre un lenguaje de programación orientado a objetos, portátil y robusto. Si al principio parecía que vendía humo, con el paso del tiempo los programadores empezaron a ver la revolución que suponía.

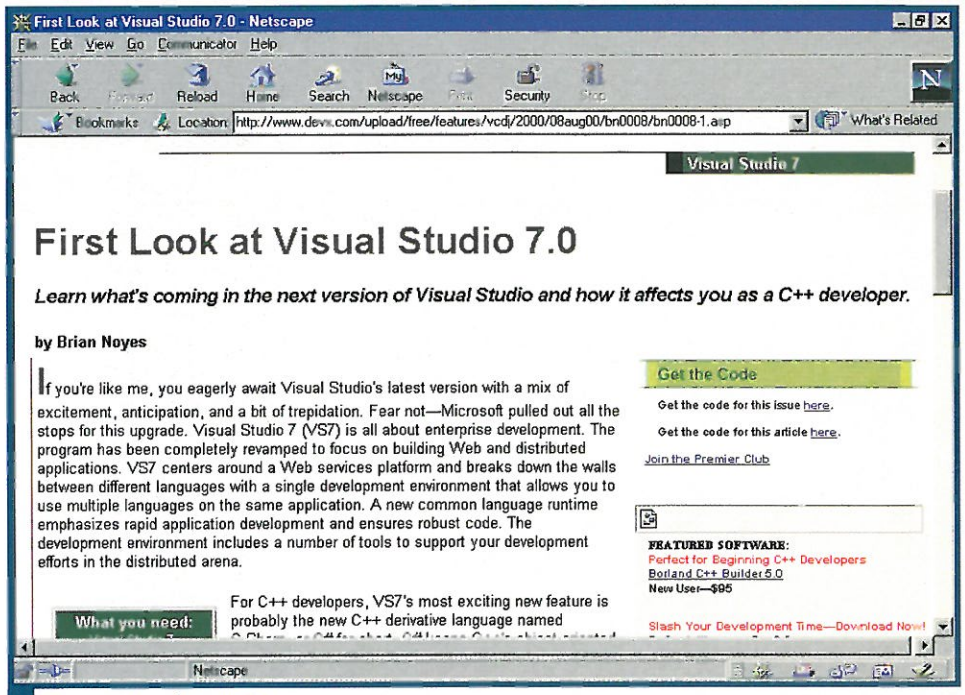
Sus armas eran la facilidad para trabajar con aplicaciones de entornos *Internet*, su portabilidad y novedad, y su orientación a objetos sin necesidad de liberar punteros -no se utilizaban-.

Con el tiempo *Java* iba madurando, y en la actualidad, el 80% de las aplicaciones se desarrollan en *Java*. La indiferencia inicial de *Microsoft* respecto a *Internet* fue decisiva para la evolución de *Java*. Mientras *Netscape* presentaba un navegador con *Java* que su usaba mediante *Applets* y se integraba en las páginas *HTML*, *Microsoft* aún no había desarrollado *Internet Explorer*.

Finalmente, sonó la campana de alarma, y los responsables de *Microsoft* rectificaron su política respecto a *Internet.* Emprendieron una vertiginosa carrera para proceder a lo que ha sido una de las adaptaciónes más impresionante de la industria informática, consiguiendo en un tiempo de récord -y con productos de calidad- rozar el monopolio en *Internet*.

En apenas tres meses, *Microsoft* lanzaba al mercado el *Internet Explorer*, comenzó a liberar una inmensa cantidad de aplicaciones para *Internet*, y cambió su estrategia para que las redes *Windows* pudieran soportar la creación de *Intranets*, el acceso rápido y sencillo a *Internet*, etcétera. Además, se hizo con los servicios de auténticos expertos del entorno *Sun*, *HP* y *Linux*, poseedores de una experiencia que resultaría vital para el éxito de la operación.

El resultado de esta carrera desenfrenada, fue que sus entornos *Windows* estaban adaptados a *Internet*, pero olvidaron crear un lenguaje de programación que facilitara la creación de aplicaciones... y mientras tanto, *Java* crecía y crecía.


First Look at Visual Studio 7.0 - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://www.devx.com/upload/free/features/vcdj/2000/08aug00/bn0008/bn0008-1.asp
What's Related
Visual Studio 7
First Look at Visual Studio 7.0
Learn what's coming in the next version of Visual Studio and how it affects you as a C++ developer.
by Brian Noyes
If you're like me, you eagerly await Visual Studio's latest version with a mix of excitement, anticipation, and a bit of trepidation. Fear not—Microsoft pulled out all the stops for this upgrade. Visual Studio 7 (VS7) is all about enterprise development. The program has been completely revamped to focus on building Web and distributed applications. VS7 centers around a Web services platform and breaks down the walls between different languages with a single development environment that allows you to use multiple languages on the same application. A new common language runtime emphasizes rapid application development and ensures robust code. The development environment includes a number of tools to support your development efforts in the distributed arena.
What you need:
For C++ developers, VS7's most exciting new feature is probably the new C++ derivative language named
Get the Code
Get the code for this issue here.
Get the code for this article here.
Join the Premier Club
FEATURED SOFTWARE:
Perfect for Beginning C++ Developers
Borland C++ Builder 5.0
New User—$95
Slash Your Development Time—Download Now!
Netscape
Figura 2. - Visual Studio 7.0.

Pero aunque pareciera un olvido, la realidad es que los desarrolladores de *Microsoft* trabajaban en ello... pero al ser una oferta de negocio tan especializada, a la hora de crear un nuevo lenguaje de las características de *Java* (la portabilidad va en contra de su propio negocio), estaban algo fuera de juego.

Durante algún tiempo circularon por *Internet* sencillos y reducidos borradores de lo que se denominaba *COOL++*. *Microsoft* daba a entender así que tenía la alternativa a *Java*. Todo el mundo esperaba que evolucionara hasta una especificación más o menos formal, y de ahí en adelante hasta alcanzar una auténtica competencia a *Java*. Pero el tiempo pasaba, y no surgía nada serio.

Los propios ingenieros de *Microsoft* no tenían claro qué pasaba. El grupo de compiladores siempre ha tenido fama de ser uno de los mejores del mundo, de hecho, cuando les dejan trabajar lo demuestran. *Visual J++(VJ++)* fue durante un tiempo el entorno con el compilador más rápido del mercado. Claro está, luego todo se fue al traste cuando rechazaron las especificaciones de *Sun* respecto a *Java* y dejaba de ser portátil y compatible.

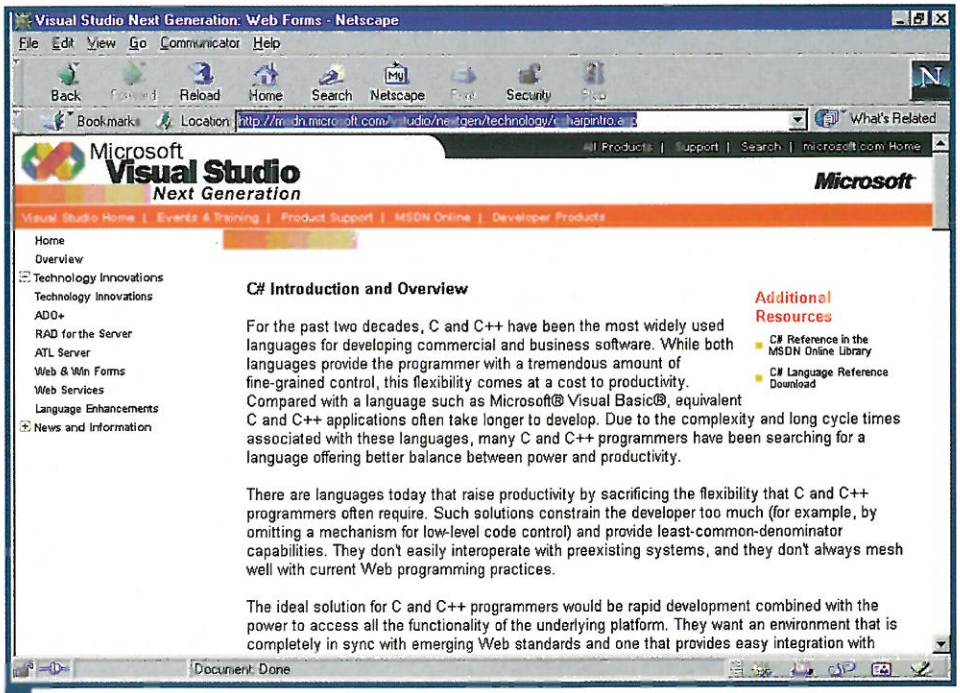
Java tampoco ha inventado nada, pero ha sido el primero en unir todo lo que los demás no conseguían. Nadie quería punteros, todos los grandes desarrolladores estaban hartos de la cacareada portabilidad del C++. *Java* presentaba solución a éstos y a otros problemas, de forma moderna, clara y verdaderamente buena.

¿Por qué no lo había hecho también *Microsoft*? ¿Por qué ha tardado tanto en presentar una mínima alternativa?. Puede que ni la propia compañía encuentre una explicación clara a semejante error.

PRESENTACIÓN DE C#

La espera y la incertidumbre acabó este mes de junio. *Microsoft* presentaba públicamente y confirmaba su decisión de lanzar al mercado un nuevo lenguaje de programación denominado *C#* (en inglés *"C sharp"*, en castellano sería... ¡*C almohadilla*!). Su creación supone una alternativa seria a *Java,* y su integración en *Visual Studio 7.0* hará que muchos desarrolladores encuentren una oportunidad de utilizar un nuevo lenguaje sin necesidad de cambiar de entorno de desarrollo, lo que es ideal (otros fabricantes deberían tomar nota).

Durante los últimos años, ninguno de los lenguajes de *Microsoft* ha sido ampliamente utilizado para resolver los grandes proyectos comerciales y de negocios. Entre los grandes entornos *mainframes* que subsisten y los maduros sistemas *Unix*, los proyectos de cierta envergadu-


Visual Studio Next Generation: Web Forms - Netscape
File Edit View Go Communicator Help
Back Reload Home Search Netscape Security
Bookmarks Location What's Related
Microsoft Visual Studio Next Generation
Microsoft
Home
Overview
Technology Innovations
Technology Innovations
ADO+
RAD for the Server
ATL Server
Web & Win Forms
Web Services
Language Enhancements
News and Information
C# Introduction and Overview
For the past two decades, C and C++ have been the most widely used languages for developing commercial and business software. While both languages provide the programmer with a tremendous amount of fine-grained control, this flexibility comes at a cost to productivity. Compared with a language such as Microsoft® Visual Basic®, equivalent C and C++ applications often take longer to develop. Due to the complexity and long cycle times associated with these languages, many C and C++ programmers have been searching for a language offering better balance between power and productivity.
There are languages today that raise productivity by sacrificing the flexibility that C and C++ programmers often require. Such solutions constrain the developer too much (for example, by omitting a mechanism for low-level code control) and provide least-common-denominator capabilities. They don't easily interoperate with preexisting systems, and they don't always mesh well with current Web programming practices.
The ideal solution for C and C++ programmers would be rapid development combined with the power to access all the functionality of the underlying platform. They want an environment that is completely in sync with emerging Web standards and one that provides easy integration with
Additional Resources
C# Reference in the MSDN Online Library
C# Language Reference Download
Document: Done
Figura 3. - VisualStudio con C#.

ra estaban fuera del alcance de *Windows*. Así, la mayoría de las grandes aplicaciones se han creado en *C* y *C++*, y en los últimos años, mediante *Java*.

Los proyectos *C/C++* tienen unos amplios y complejos ciclos de desarrollo, por lo que hay que buscar un término medio entre esto y la productividad. El objetivo es conseguir los tiempos y costes de desarrollo para aplicaciones desarrolladas con *VB* y *VC++*, con las posibilidades de flexibilidad, robustez y mantenimiento de *C/C++*. Además, hay que ofrecer un producto que permita desarrollar aplicaciones *Web*, mejorando la falta de integración que tienen *C/C++*, debido a la necesidad de un desarrollo muy específico y conocimientos altos.

CARACTERÍSTICAS

C# se presenta como un lenguaje moderno y orientado a objetos que permitirá a los desarrolladores construir muchas aplicaciones de forma rápida y robusta, dentro de la nueva plataforma *.NET* definida por *Microsoft*, que ofrece un conjunto de herramientas y servicios que permiten el control total en comunicaciones, cálculo y desarrollo, por lo que determinadas tareas reiterativas y tediosas no volverán a programarse.

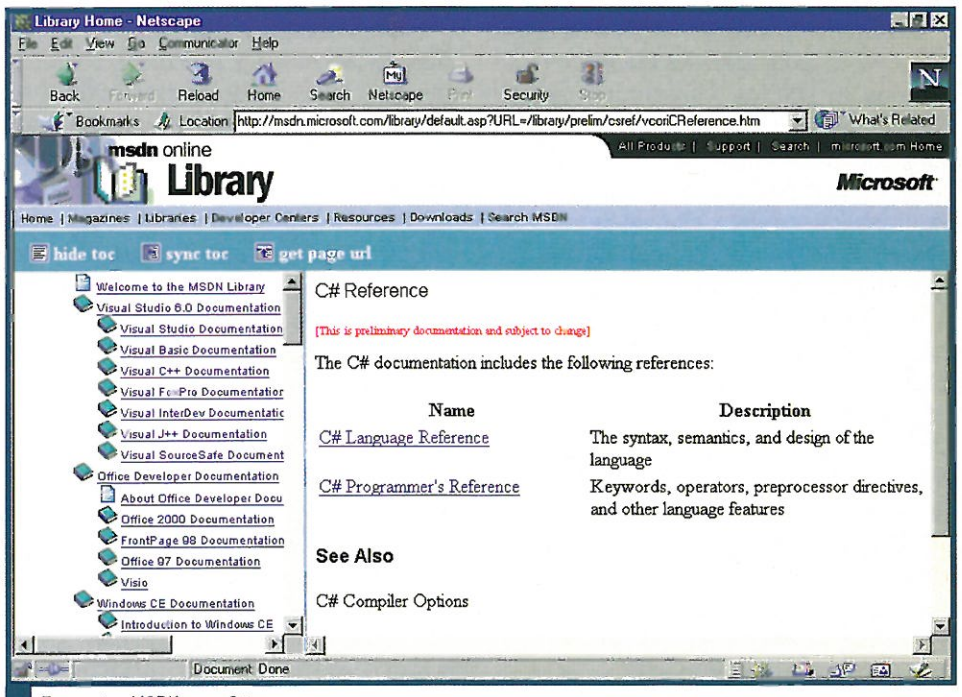
Gracias a su orientación a objetos, *C#* se podrá utilizar en arquitecturas que requieran un gran conjunto de componentes, desde aplicaciones de negocio desarrolladas a alto nivel hasta aplicaciones de sistemas con acceso a bajo nivel. Gracias al diseño de *C#*, los componentes desarrollados pueden ser accesibles a través del *Web*, utilizados desde cualquier parte y desde cualquier sistema operativo -en realidad es una interfaz perfectamente definida de acceso a sus servicios web-.

Otra de las características de *C#* es que está pensado para obtener unos tiempos de desarrollo mucho más rápidos que si utilizáramos entornos *C++* tradicionales, pero con unos rendimientos muy similares. Además,los desarrolladores necesitan un periodo de aprendizaje muy corto antes de empezar a usarlo.

PRODUCTIVIDAD Y SEGURIDAD

Los cambios producido por *Internet* en las comunicaciones, economía, forma de vida, etcétera, también han afectado a la forma en la que se desarrolla el software. Los ciclos de desarrollo son cada vez más cortos, pues la competencia cada vez es más rápida y presenta productos de calidad a menor coste. *Internet* ha permitido a los grandes constructores de software poner en contacto a sus diferentes grupos de desarrollo en distintos lugares del mundo, mejorando así su forma su actividad.

Por ello, una de las prioridades en la confección de *C#*, fue que soportara y permitiera la creación de software en ciclos cortos, pero permitiendo un mantenimiento de calidad (por ejemplo, con revisiones incrementales), más que ser una simple y monumental nueva versión. Así se busca reducir el número de líneas de código


Library Home - Netscape
File Edit View Go Communicator Help
Back Reload Home Search Netscape Security
Bookmarks Location http://msdn.microsoft.com/library/default.asp?URL=/library/prelim/csref/vcoriCReference.htm What's Related
msdn online Library
Microsoft
Home | Magazines | Libraries | Developer Centers | Resources | Downloads | Search MSDN
hide toc sync toc get page url
Welcome to the MSDN Library
Visual Studio 6.0 Documentation
Visual Studio Documentation
Visual Basic Documentation
Visual C++ Documentation
Visual FoxPro Documentation
Visual InterDev Documentation
Visual J++ Documentation
Visual SourceSafe Document
Office Developer Documentation
About Office Developer Docu
Office 2000 Documentation
FrontPage 98 Documentation
Office 97 Documentation
Visio
Windows CE Documentation
Introduction to Windows CE
C# Reference
[This is preliminary documentation and subject to change]
The C# documentation includes the following references:
Name
Description
C# Language Reference
The syntax, semantics, and design of the language
C# Programmer's Reference
Keywords, operators, preprocessor directives, and other language features
See Also
C# Compiler Options
Document: Done
Figura 4. - MSDN para C#.

generadas, restringiendo la probabilidad de error.

NUEVAS POSIBILIDADES WEB

C# ofrece un nuevo modelo en el desarrollo de aplicaciones, basado en ofrecer más y mejores soluciones en la adaptación y uso de los nuevos y continuos estándares que surgen para el *Web*, desde el *HTML* y el *XML*, hasta el *SOAP*. La idea es ofrecer un producto que se adapte a las necesidades actuales, ya que según *Microsoft*, las herramientas de desarrollo están pensadas y basadas en antiguos modelos para el *Web* que no se adaptan a las necesidades actuales por los múltiples cambios que se suceden en *Internet*.

Microsoft carecía de un lenguaje adaptado a Internet

Con *C#* se podrá acceder a las posibilidades que ofrece *.NET*. Se posibilita la adaptación de los distintos componentes para que sean utilizados en el *Web*, e incluso se podrán añadir componentes a la misma plataforma de forma que ésta se adapte y personalice según las necesidades del desarrollo.

Pero *C#* esta pensado no sólo para poder utilizar el *Web* de forma cómoda y sencilla, sino para incluso construir aplicaciones que pueden competir con otras similares. Por ejemplo, al utilizar *XML* para comunicar diferentes componentes a través de *Internet*, lo que se suele hacer es *mapear* los datos a una clase. Esto puede ser algo costoso en rendimiento, pero mediante el uso de *C#* se puede *mapear* a una estructura, de forma que el rendimiento obtenido es claramente superior.

ERRORES

Los programadores de *C++* reconocerán lo fácil que es cometer errores en un simple despiste. Pregunten a los programadores en *Corba* con *C++* si se pueden llegar a cometer errores antes de alcanzar un determinado nivel de experiencia. Los clientes exigen software de calidad muy superior a la que se pedía hace años. La competencia presenta productos baratos y muy buenos, y esto obliga a que no se pueda vender software con deficiencias claras. ¿Por qué aún usamos *Windows*?

En lo referente a la escritura de programas en *C#*, muchos de los errores típicos son vigilados. El inicio de variables, tipos seguros y el recolector de basura, son algunas de las características que permiten desarrollar mediante este

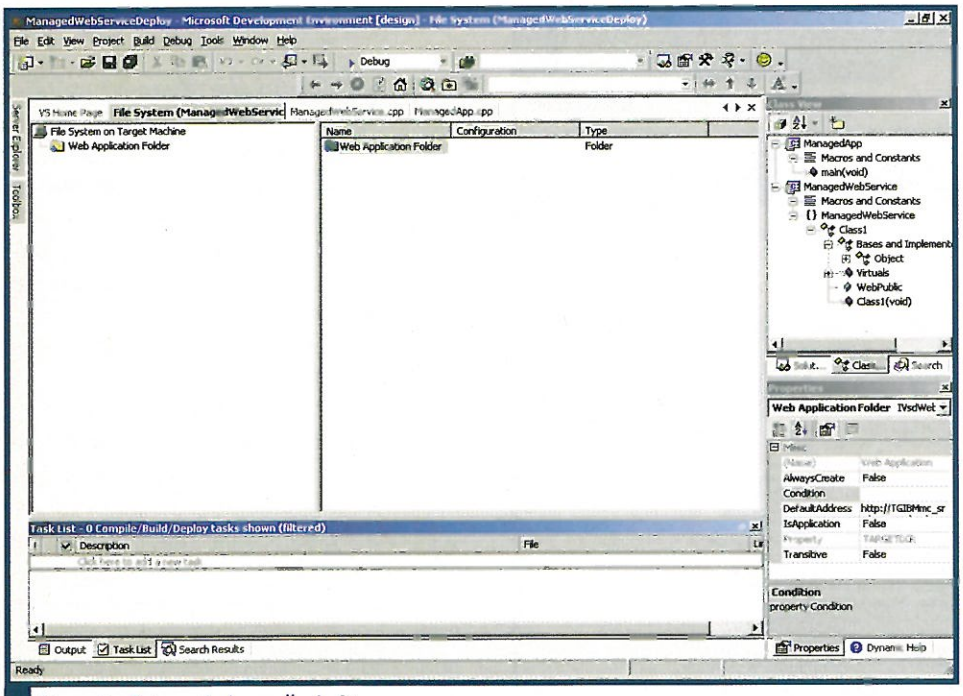


Figura 5. - Entorno de desarrollo de C#.

S.U.S.E LINUX
¡¡NUEVA VERSION 7.0 OFICIAL!!
1.500 Aplicaciones · 6 CD Roms
SuSE
Internet/Intranet
Multimedia
Desarrollo
Seguridad
ReiserFS
StarOffice 5.2
4 Manuales
NUEVO
SuSE LINUX 7.0
Edición en español
60 días de asistencia técnica gratuita para la instalación
7.069 pts.
Precio de venta recomendado IVA no incluido
Versión 7.0 completa y oficial
6 CD ROM's, +1.500 aplicaciones
¡Ahora con 4 Manuales en castellano!
instalación y ayuda en castellano
Caja y presentación en castellano
con Word Perfect 8 y Star Office 5.2
Soporte por e-mail durante 60 días
MAYORISTA OFICIAL PARA ESPAÑA
SuSE
Precios especiales a distribuidores y cadenas de tiendas según cantidades
¡Consúltenos!
¡¡Visite nuestro Web!!
http://www.abcnet.es/linux
30 años 1970-2000
91 634 20 00
abc analog, s.l. FAX 91 634 47 86

lenguaje "fuertemente *tipado*", dando más importancia a la lógica en sí que a la forma en la que se expresa, permitiéndose así resolver problemas complejos con mínimas garantías. Se han eliminado las múltiples formas de hacer la misma tarea en lenguajes como *C/C++*, lo que producirá un código más claro.

COSTES DE DESARROLLO

Según *Microsoft*, siguiendo algunos métodos de desarrollo definidos actuales, los tiempos y costes derivados del análisis de requisitos, planificación, etcétera, son muy elevados, y si añadimos a ello que desde la resolución de éstos hasta que se realiza la implementación surgen múltiples cambios y revisiones, el resultado es un incremento en los costes estimados.

Hoy por hoy, las herramientas de desarrollo no permiten la continuidad entre la lógica de negocio y el código, por lo que los desarrolladores tienden a preparar código y más código lleno de comentarios como posibles parches para esa lógica de negocio que no se conoce o está a punto de definirse, pero que se necesita para crear otras partes del desarrollo.

Para resolver algunos de estos problemas, con *C#* se puede crear un conjunto de *metadatos* que definen los atributos propios del dominio en el que trabajamos y aplicarlos posteriormente en la creación de clases, interfaces o similares. Gracias a esto, el desarrollador puede gestionar cada elemento asegurándose de que habrá una correspondencia entre la parte abstracta del negocio definido y la implementaron construida. El uso de *metadatos* permite este proceso, ayudando a la unión de estos dos elementos.

Bien, a mi entender es más o menos correcta la idea de *Microsoft*, aunque algo exagerada. Respecto a los costes iniciales de desarrollo, es cierto si se siguen las "modernas" técnicas de análisis y diseño, como por ejemplo *UML*, pero como resultado se obtiene un menor tiempo en el etapa de desarrollo con un número mucho más reducido de errores y posibles interpretaciones erróneas desde la lógica de negocio a la implementación. Que en la actualidad no haya herramientas que unan estas etapas no es del todo cierto, aunque sí lo es que no permiten una cierta validación en ambos sentidos o una simulación necesaria cuando no hay definidos determinados requerimientos.

PORTABILIDAD

Microsoft defiende su entorno y no cree en la portabilidad, alegando que las experiencias reales demuestran que tarde o temprano se tienen que hacer operaciones de acceso "nativo" para mejorar el funcionamiento o calidad de las aplicaciones. Por tanto, *C#* no define la portabilidad entre diferentes sistemas operativos.

Como alternativa, se entiende la portabilidad como la posibilidad de crear aplicaciones de calidad de forma distribuida basadas en las posibilidades de *C#* para la creación de objetos *COM* y acceso basado en los *APIS* propios de *Windows*.

Como cada objeto en *C#* corresponderá con un objeto *COM* (entendido como servicio), el desarrollador no tendrá que definir ni *interfaces COM* ni similares, incluso se podrá acceder a punteros nativos con determinadas restricciones, potenciando así el rendimiento de muchas aplicaciones. Es más, los desarrolladores aún podrán seguir manejando la memoria, pero eso sí, ahora de una forma segura, mediante el uso de funciones *inline*.

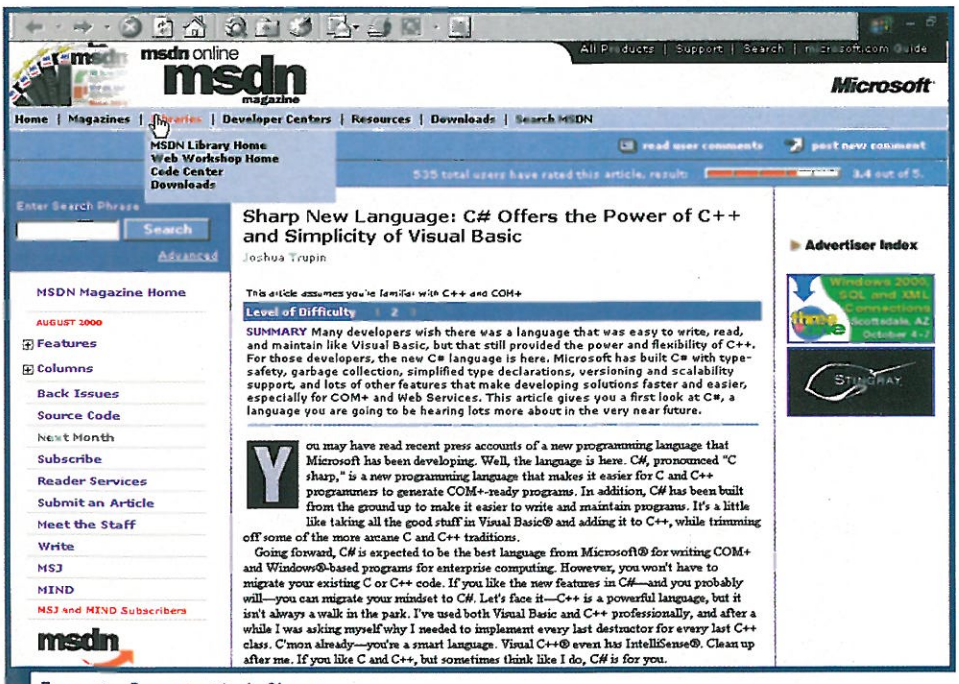
ESTANDARIZACIÓN

Microsoft ha tenido un gran acierto al proponer que se cree una especificación para *C#* por una organización ajena al propio fabricante, como es la *ECMA* (*European Computer Manufactures Association*). El objetivo es que *ECMA* realice todas las tareas de estandarización para la *ISO* (*International Standards Organization*).

Con C# se mejora la productividad de las aplicaciones

En realidad, lo que *Microsoft* pretende con esta medida, es que diferentes constructores de software creen muchas adaptaciones de *C#* en diferentes plataformas y sistemas operativos, y esto, llegado el momento, lleve a la especificación única.En el poco tiempo que lleva la especificación, muchos fabricantes han tomado ya partido. Empresas como *devSoft*, *NuMega* y *Rational*, ya forman parte de este proyecto.

Ya se ha definido qué es el lenguaje C#, también se han detallado sus antecedentes, descrito las principales características y mostrado cómo se intenta adaptar a las nuevas necesidades basadas en un futuro no muy lejano. Nos queda la parte más interesante para los desarrolladores, y es ver qué forma tienen los programas escritos en C#... y ese será el asunto del próximo artículo.


msdn online
msdn magazine
Microsoft
Home | Magazines | Libraries | Developer Centers | Resources | Downloads | Search MSDN
MSDN Library Home
Web Workshop Home
Code Center
Downloads
Sharp New Language: C# Offers the Power of C++ and Simplicity of Visual Basic
Joshua Trupin
Level of Difficulty
SUMMARY Many developers wish there was a language that was easy to write, read, and maintain like Visual Basic, but that still provided the power and flexibility of C++. For those developers, the new C# language is here. Microsoft has built C# with type-safety, garbage collection, simplified type declarations, versioning and scalability support, and lots of other features that make developing solutions faster and easier, especially for COM+ and Web Services. This article gives you a first look at C#, a language you are going to be hearing lots more about in the very near future.
You may have read recent press accounts of a new programming language that Microsoft has been developing. Well, the language is here. C#, pronounced "C sharp," is a new programming language that makes it easier for C and C++ programmers to generate COM+-ready programs. In addition, C# has been built from the ground up to make it easier to write and maintain programs. It's a little like taking all the good stuff in Visual Basic® and adding it to C++, while trimming off some of the more arcane C and C++ traditions.
Going forward, C# is expected to be the best language from Microsoft® for writing COM+ and Windows®-based programs for enterprise computing. However, you won't have to migrate your existing C or C++ code. If you like the new features in C#—and you probably will—you can migrate your mindset to C#. Let's face it—C++ is a powerful language, but it isn't always a walk in the park. I've used both Visual Basic and C++ professionally, and after a while I was asking myself why I needed to implement every last destructor for every last C++ class. C'mon already—you're a smart language. Visual C++® even has IntelliSense®. Clean up after me. If you like C and C++, but sometimes think like I do, C# is for you.
MSDN Magazine Home
Features
Columns
Back Issues
Source Code
Next Month
Subscribe
Reader Services
Submit an Article
Meet the Staff
Write
MSJ
MIND
MSJ and MIND Subscribers
Search
Advertiser Index
Figura 6. - Presentación de Sharp.

HASTA UN 45% DE DESCUENTO

PREPÁRATE PARA SER EL MEJOR WEBMASTER

- ✓ En España se ha duplicado el número de usuarios en los últimos 8 meses.
  Fuente: AIMC
- ✓ El 80% de internautas existentes en todo el mundo entran en la Red para realizar negocios.
  Fuente: FORRESTER RESEARCH
- ✓ 100 millones de personas navegan hoy.
  Fuente: Varios compilado por NUA INTERNET SURVEYS
- ✓ En 5 años el comercio electrónico supondrá 4,5 billones de pesetas tan sólo en los Estados Unidos.
  Fuente: IBID
- ✓ En el año 2.000 la gran mayoría de las transaciones no realizadas en dinero metálico serán mediante pagos electrónicos.
  Fuente: KILLEN ASSOCIATES

BOLETÍN DE SUSCRIPCIÓN

OFERTA VÁLIDA SÓLO PARA ESPAÑA

SI, deseo suscribirme al **CURSO WEBMASTER DE PROGRAMACIÓN INTERNET**

CONDICIONES DE PAGO (MARQUE LA OPCIÓN DESEADA)

- ❑ 49 fascículos a 750 Ptas.: **TOTAL 36.750 PTAS.**
- ❑ 4 carpetas para archivar fascículos a 950 Ptas.: **TOTAL 3.800 PTAS.**
- ❑ 1 carpeta para archivar disquetes y CD´s a 1.250 Ptas.
- ❑ Se regalará la versión de **FrontPage 2000** en el primer envío valorada en **29.750 Ptas.**

PRECIO TOTAL DE LA COLECCIÓN: 71.500 PTAS.

PAGO AL CONTADO:

- ❑ Ahorro del **45%. TOTAL PAGO AL CONTADO: 39.325 Ptas.**

PAGOS APLAZADOS:

- ❑ Pago en 3 entregas ahorro del **38%**. Cada entrega **14.780 Ptas.**
- ❑ Pago en 6 entregas ahorro del **30%**. Cada entrega **8.340 Ptas.**
- ❑ Pago en 12 entregas ahorro del **20%**. Cada entrega **4.770 Ptas.**

ENTREGA DEL FRONTPAGE 2000 EN LA ÚLTIMA ENTREGA

UTILICE MAYÚSCULAS PARA RELLENAR EL PEDIDO

Nombre y apellidos .....F. nacimiento.....

Domicilio ..... C.P. .....

Ciudad ..... Provincia .....Telf. .....

e-mail ..... Profesión .....

FORMAS DE PAGO:

❑ Con **cargo a mi tarjeta VISA** Nº ____/____ /____ /____ Caduca .....

❑ **Domiciliación bancaria**

Ruego a Uds. que se sirva cargar en mi ❑ cuenta corriente ❑ libreta de ahorro el/los recibos que le serán presentados por REVISTAS PROFESIONALES, S.L. como pago de mi suscripción al **CURSO WEBMASTER DE PROGRAMACIÓN INTERNET**

CÓDIGO CUENTA CORRIENTE

ENTIDAD	OFICINA	DC	Nº CUENTA

❑ **Cheque** a nombre de REVISTAS PROFESIONALES, SL.

❑ **Contra reembolso** del importe más gastos de envío.

❑ **Giro Postal** (adjunto fotocopia del resguardo)*.

*Esta forma de pago es exclusivamente para pagos al contado

Firma

Los increíbles botones Web de Flash 4

Antoni Masso Mola.

La creación de botones es una parte fundamental en el diseño y funcionamiento de una página *web*. En este artículo aprenderás a hacer que tus botones presenten increíbles efectos sobre los navegadores y presentaciones.

Macromedia Flash 4 es un potente programa basado en vectores para realizar páginas web, con la posibilidad de incluir efectos y diseños que sin duda alguna sería imposible conseguir con cualquier otra herramienta o lenguaje de programación.

En este primer artículo trataremos la utilización de botones en *Flah 4*, pero desde una perspectiva más avanzada, nos saltaremos el proceso de creación de un botón regular y nos centraremos en aquellos que ofrecen una imagen de gran diseño a nuestra página web. Echaremos un vistazo a la interacción entre gráficos-botones y *movies*-botones.

Con *Flash 4*, todo botón será único, pues sus posibilidades sólo se limitan a la imaginación del lector. Siguiendo este manual, el lector podrá realizar un sin fin de botonesde efectos únicos.

BOTONES CON EFECTOS ON-MOUSE

Veamos pues, la creación de botones que obedezcan las funciones *OnMouseUp*, *OnMouseOver* y *OnMouseDown*. Como se puede ver en pantalla, una vez en la zona de edición del botón, el entorno es muy sencillo. Esto puede hacer que a más de a uno le parezca una tarea muy sencilla crear un botón. No nos equivoquemos, ya que por culpa de un pequeño error, el botón ya no seguirá las pautas marcadas.

El escenario de edición de botones es muy sencillo (Figura 1). En este caso estamos editando el botón "Sólo Programadores 1". *Flash 4* crea automáticamente un *keyframe* para introducir los gráficos que se deseen. El primer *frame* es el llamado *Up*. Es el que aparece es-

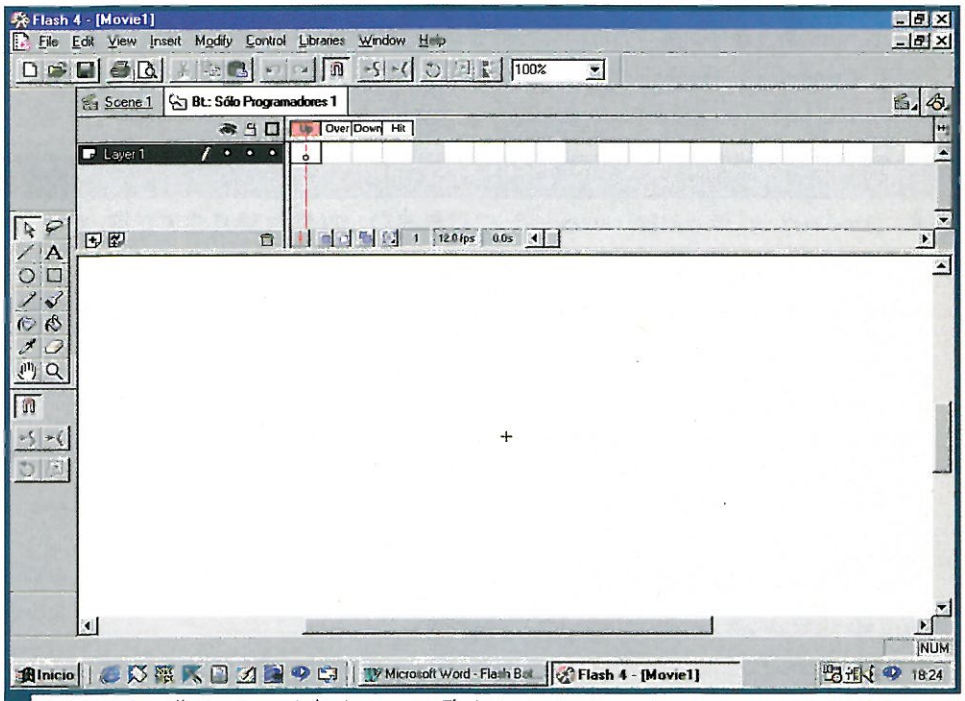


Figura 1. - Pantalla de edición de los botones en *Flash 4*.

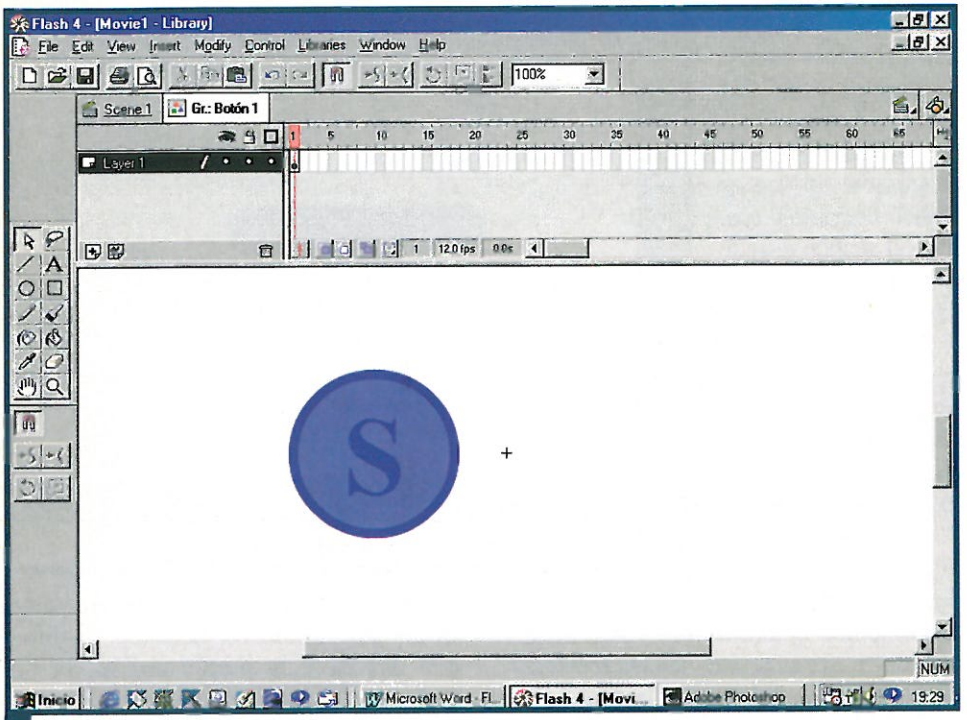


Figura 2. -Diseño del gráfico principal.

tático en pantalla (si no se le ha incluido ningún efecto o animación) hasta que el ratón actúa sobre él.

El primer botón que crearemos usará la acción *OnMouseOver* y una animación. La acción *OnMouseOver* sólo da la orden de cambiar de marco cuando el ratón está sobre el botón. Por ejemplo, si creamos un botón con un círculo rojo en el *keyframe OnMouseUp* y otro verde en *OnMouseOver*, cuando el ratón se sitúe sobre el botón, éste pasará de rojo a verde. Mientras el ratón esté encima del botón, el círculo será de color verde, y al quitar el puntero, volverá a su color original.

Primero tenemos que diseñar la animación y los gráficos. El primer paso es insertar un nuevo símbolo, en este caso, un gráfico. El gráfico constará de un círculo con la letra *S* en su interior. Éste será introducido en el *keyframe* de *OnMouseUp*. Para crear el gráfico, sólo hay que elegir seleccionar la herramienta de círculo, apareciendo una barra con 4 opciones la parte inferior: color del borde del círculo, tamaño del borde, estilo del borde, y fondo del interior del círculo. Seleccionaremos el color azul marino como borde del círculo. El borde lo pondremos a 8.0 puntos. El estilo del borde lo dejaremos como *Solid*. Por último, el fondo del interior del círculo será azul claro.

Una vez seleccionadas las opciones para la creación del círculo, únicamente deberemos crearlo. Las opciones seleccionadas aparecerán automáticamente. Si el tamaño del círculo no es el deseado, una vez en el escenario de la página, se puede seleccionar el círculo, y con el botón derecho del ratón escoger *Scale* para aumentar o reducir el tamaño.

Con Flash 4, los botones son únicos

El gráfico ya está casi terminado. Sólo falta introducir el texto, (*S*). Para ello, hay que seleccionar la herramienta de texto, la casilla cuadrada con una *A* en su interior. Escogeremos la fuente *Times New Roman* 24. El color de la letra será el mismo que el borde del círculo, azul marino. Seleccionamos la opción *Bold* para hacer que la S de Sólo Programadores resalte más que el resto. Ahora sólo debemos presionar sobre el centro del círculo e introducir la letra *S*. Una vez hecho esto, y según el tamaño y situación del círculo, deberemos mover el texto o aplicar la opción *scale*, para que quede centrado y con un tamaño visible, como explicamo más arriba. El círculo ya ha sido creado, y está en el *keyframe* 1 del símbolo *Gr.: Botón 1*. Este símbolo está guardado en la librería de objetos creados de la presentación en la que estamos trabajando. Para ver la librería y poder acceder a los símbolos en ella presentes, se debe acceder a *Window*, y luego entrar en *Library*. Una ventana *Pop-up* aparecerá. En su interior tendremos guardados los símbolos *Bt.: Sólo Programadores* y *Gr.: Botón 1*. Con un doble clic sobre uno de ellos, accederemos a su editor.

El siguiente paso es más complicado, se trata de crear una animación para introducir en la acción *OnMouseOver* del botón. Como hemos realizado con los anteriores símbolos, primero crearemos uno nuevo, pero esta vez con la propiedad *movie*. Lo llamaremos *Movie Botón 1*. Para que funcione correctamente, es imprescindible prestar atención y seguir los pasos correctamente. La primera acción será crear un rectángulo que termine en un semicírculo.

Para ello iremos a *Insert* y seleccionaremos *New Symbol*. En la nueva pantalla teclearemos el nombre de la animación y haremos clic sobre *Movie Clip*. Para realizar el gráfico principal, se puede crear un rectángulo con la opción cuadrada en la barra de herramientas. La longitud deberá ser la suficiente para introducir el texto *Programadores* con fuente 24. Una vez creado el rectángulo debemos crear un círculo. Con el ratón seleccionamos una mitad vertical del círculo y la eliminamos. Finalmente movemos el medio círculo y lo insertamos en el rectángulo, de manera que los bordes horizontales encajen perfectamente.

El color de fondo debe ser un azul claro, parecido al del interior del círculo. Una vez tengamos el rectángulo listo podremos introducir el texto *Programadores* dentro de él. No usaremos el estilo *Bold*, para diferenciar el botón principal con la acción secundaria.

En el último paso simularemos que el rectángulo aparece de la nada, para lo que crearemos un *keyframe* cada dos *frames* hasta llegar al número 10, (5 *keyframes* en total). Ahora debemos hacer que la animación muestre al rectángulo formándose cuando la animación es activada. Para realizar tal tarea, deberemos centrarnos en los cinco *keyframes*, creados desde el *Layer 1* y haciendo clic con el botón derecho del ratón en el primer marco. Escogeremos la opción *Insert Keyframe*, y repetiremos el proceso cinco veces hasta disponer de cinco marcos con la misma imagen.

Cada uno de ellos incluye todo el gráfico. El último *keyframe* es el que deberá quedar como el gráfico completo. Del primer al cuarto *keyframe*, el rectángulo deberá crecer. Para lograrlo, deberemos asegurarnos que en cada *keyframe* hemos dado la orden *Break Apart* para cada gráfico. Para ello, hay que seleccionar el elemento e ir a *Modify* y *Break Apart*. Lo que se consigue con esta herramienta es hacer que cada píxel sea independiente del resto y así poder editarlo en su totalidad.

Ahora podemos realizar el corte de gráfico para cada *keyframe*. En el primero de ellos, seleccionaremos desde el semicírculo hasta la primera *r* de *programadores*. Lo situaremos exactamente al principio del rectángulo con el fin que ocupe exactamente el lugar del gráfico anterior, para que éste no pueda ser visible. Haremos estas divisiones sucesivamente hasta llegar al quinto *keyframe,* donde el gráfico no debe modificarse, pues es el resultado final. Por último, se introduce la acción *Stop* en el último *keyframe* del *movie*, haciendo doble clic sobre el marco 10. Accederemos a la sección *Actions* y presionaremos sobre el botón con la señal +. En ella seleccionaremos *Stop* y aceptaremos. Hay que incluir la orden *Stop* al final de la animación, pues en caso contrario, la película se repetiría indefinidamente al llegar al final de los *frames*. Siempre que se desee, se puede abrir el fichero *boton1.fla* encontrado en el *CD-ROM* que acompaña a la revista.

Con Scale, cambiamos el tamaño del círculo

El último paso para hacer que el gráfico y el *movie* funcionen como un único botón, es unirlos en el símbolo que anteriormente creamos denominado *Botón Sólo Programadores*. Una vez dentro del editor del botón, seleccionaremos el *keyframe* donde reside la acción *OnMouseUp* e insertaremos el gráfico. Este último paso lo conseguiremos abriendo la librería e insertando el gráfico deseado, en nuestro caso *Gr.: Botón 1*.

Una vez tengamos el gráfico en el *keyframe* principal, introduciremos otro sobre la acción *OnMouseOver*. Lo que conseguimos con esto, es situar el gráfico en ambas acciones para que sea visible cuando el puntero esté o no sobre el botón. Antes de continuar hay que asegurarse de que en ambos *keyframes* aparece el gráfico exactamente en el mismo lugar.

Para terminar con la creación del botón, nos situamos sobre *OnMouseOver* e introducimos el símbolo *Mv.: Movie Botón 1*. Para ello, deberemos abrir la librería donde residen los objetos creados. Seleccionamos la animación, y con el ratón la metemos en *OnMouseOver*. Ahora nos aseguraremos -con *Scale*- de que ambos símbolos tienen la misma anchura. Finalmente, seleccionamos la película e iremos a *Modify*, *Arrange* y *Send to Back*, con el fin de hacer que la animación ocupe la segunda capa, dando prioridad de visión al gráfico.

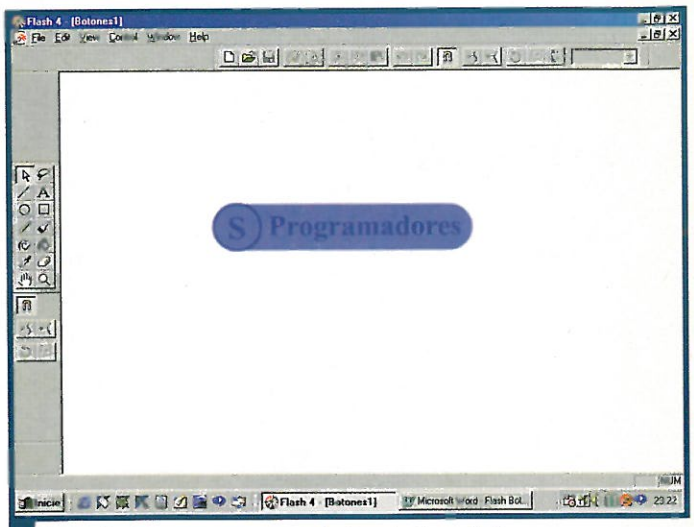


Figura 3. -Aspecto que presenta el primer botón acabado.

El botón puede ser usado de la misma manera, pero con la función *OnMouseDown*. El funcionamiento del botón sería el mismo, pero en vez de presentar el rectángulo despegable cuando el puntero pasa por encima de él, lo hará al deslizarse fuera de él.

La animación de este botón siempre realzará el diseño y calidad de cualquier página estática. Ahora, editar y crear otras animaciones y gráficos es verdaderamente sencillo.

ANIMACIONES COMPLEJAS

Flash 4 ofrece tantas posibilidades para crear botones, que sabiendo introducir películas dentro de un botón, se pueden conseguir efectos como los de las páginas *web* más complejas. Un buen botón es el que con el mínimo espacio, más información o más botones ofrece. Este será el segundo botón avanzado del que nos ocuparemos. Se trata de hacer un botón que muestra más información a medida que seleccionas opciónes. Utilizaremos la técnica de insertar gráficos dentro de diferentes *movies*. Una vez hechos los botones y animaciones necesarias, el resultado es una mezcla que permite situar los primeros botones visibles que inician una simple acción, la de abrir otra animación en la que se encuentran los gráficos para acceder al contenido deseado.

El primer paso el diseño del botón-*movie* principal. Crearemos un nuevo símbolo llamado *Mv.: Movie 1 Botón 2* con la propiedad *movie*. En ella crearemos un rectángulo, pero esta vez utilizaremos una nueva herramienta. En el menú de herramientas que aparecen cuando se selecciona el botón de creación de rectángulos, un nuevo grupo con cinco nuevas opciones aparece al final. En este caso, elegiremos el color de borde y fondo que deseemos. Eso sí, al último botón, que es una media curva le pondremos la propiedad 10, con lo que haremos que el rectángulo tenga una curva.

En este instante ya tenemos el *movie* donde introduciremos los botones principales. Crearemos dos botones, *Capítulo 1* y *Capítulo 2*. Como ya sabemos hacer, crearemos dos nuevos símbolos y diseñaremos los botones. Deberíamos hacer que siguieran el mismo estilo rectangular que la animación, para lo que seleccionamos la herramienta necesaria y trabajamos sobre el *keyframe* principal, *OnMouseUp*.

Una vez hecho el diseño, crearemos un *keyframe* en *OnMouseDown* que será el encargado de abrir una nueva "ventana flotante", donde la nueva animación aparecerá con sus respectivos botones. En el segundo *keyframe* deberíamos cambiar el diseño del botón, para que el usuario sepa que está sobre él. Una buena idea es invertir los colores de fondo y

borde o del texto. Para lograr este resultado, utilizaremos la herramienta *Paint Bucket*, seleccionando el color e insertándolo sobre la parte del gráfico que se desee modificar. Seguiremos los mismos pasos con el segundo botón, únicamente se diferenciarán por el texto de su interior.

Tenemos la animación principal donde se hallarán los botones principales también creados, y sólo nos falta crear los gráficos de opción de *Capítulo 1* y *2* y la animación en donde se encontrarán resguardados. Para ello, crearemos un gráfico en cada uno de los botones principales. Para el *Capítulo 1* y *Capítulo 2*, introducimos un texto cualquiera, sólo para ver cómo queda en pantalla. Los diseñaremos siguiendo el mismo estilo que en los creados anteriormente. Hay una manera de duplicar cualquier símbolo de la presentación *Flash*. Se consigue abriendo la venta *Library*, se selecciona el que se desea duplicar, y con el botón derecho hacemos clicsobre él. Entre las diversas opciones, elegimos *Duplicate*. Le asignamos un nuevo nombre, y automáticamente dispondremos del mismo símbolo, que podremos editar fácilmente sin necesidad de crear uno nuevo.

Si todo ha salido correctamente, dispondremos de dos botones, dos gráficos y una animación principal. Tan sólo nos falta diseñar dos animaciones más, y ya dispondremos de todos los símbolos necesarios. Como hemos hecho antes, crearemos los *movies* con un diseño parecido al principal, pero de tamaño reducido y diferentes colores de fondo y borde. Para crearlo debemos seguir los pasos descritos más atrás, insertando un nuevo símbolo y dándole forma con las herramientas disponibles.

La orden Stop evita que la película se repita automáticamente

Sólo nos queda juntar las piezas para hacer que este botón multi funcional esté listo para ser incluido en una página web. Lo primero será juntar los gráficos en las animaciones secundarias. Abrimos el editor de uno de las animaciones, y una vez preparados para agregar los símbolos exteriores necesarios, nos aseguramos de que tenemos la librería a la vista. En pantalla tendremos el *movie* sin ningún otro elemento dentro de él. Para insertar los gráficos, seleccionamos aquél que corresponda con la animación que estamos editando. Primero crearemos un nuevo *layer* donde guardaremos los gráficos y las acciones, situándonos en el primer *layer* y haciendo clic con el botón derecho. Escogemos la opción *Insert Layer* y automáticamente tendremos dos *layers*. Una vez tengamos el gráfico en pantalla, crearemos un segundo *keyframe* en el *frame* 15. Haremos doble clic sobre el gráfico del primer *keyframe*, y accederemos a *Color Effect*. Seleccionaremos *Alpha* y lo pondremos a 0%. Tras aceptar, haremos doble clic en el primer keyframe y entraremos en *Tweening*, seleccionaremos *Motion* y presionaremos *Aceptar*. En el último *keyframe* le añadimos la acción *Stop*. Estos últimos pasos hacen que el gráfico situado en el *movie* aparezca lentamente hasta que sea completamente visible.

Haremos lo mismo para la segunda animación, con el fin de tener listas las ventanas que abrirán los botones principales. Acto seguido deberemos incluir las acciones que estos botones harán al ser utilizados. Para introducir una acción en cualquier botón, se debe hacer doble clic en el deseado. Una vez hecho esto, una ventana de opciones flotante aparecerá. Seleccionaremos *Actions*, y dentro de ella abrimos la señal de más (+) para seleccionar la opción *Get URL*. La caja de texto de la derecha es donde teclearemos la dirección a la que el botón debe dirigir al usuario. Es tan sencillo como seguir los tres pasos y aceptar. El botón ya está listo para ser utilizado. Crearemos las acciones necesarias para cada botón antes de seguir con la formación del principal.

Cuando ya estén introducidas todas las direcciones *web* deseadas para cada botón, solamente quedará introducir las animaciones secundarias en los botones principales. Nos situamos sobre el editor del botón *Capítulo 1* en el *keyframe OnMouseOver*. Para el primer botón, introduciremos el *movie* en su lado derecho. Para conseguirlo. seleccionamos el *movie* de la librería y lo introducimos en el *keyframe* deseado del botón. Haremos exactamente el mismo procedimiento para el siguiente botón, pero introduciendo el *movie* al lado opuesto.

Por último, sólo queda introducir los botones principales *Capítulo 1* y *Capítulo 2* en la animación principal. Para lograrlo seguiremos los mismos pasos anteriormente realizados, con lo que ya tenemos el botón con la acción de abrir ventana de texto con animación.

Todos los gráficos, diseños, animaciones, etcétera, se pueden cambiar para hacer una variedad infinita de botones. También es posible activar sonidos cuando el ratón pase sobre el botón, al hacer clic, etcétera. Para ello, se hace doble clic en el *keyframe* del botón deseado y seleccionamos *Sound*. En la librería debe haber algún fichero *wav* para que *Flash* detecte el sonido. Una vez elegido el fichero sonoro, se puede editar.

CONCLUSIÓN

Hemos aprendido a hacer botones que dispongan de *movies* que a su vez tengan animaciones de gráficos. Saber crear este tipo de botones no es sencillo, pero aportan un aire de gran diseño a cualquier página web. Saber crear botones vistosos con *Flash 4* es muy importante. Muchas solicitan disponer de páginas con botones con otras funciones a parte de servir como link. Con un poco de práctica, se pueden conseguir verdaderas maravillas de Arte Electrónico.

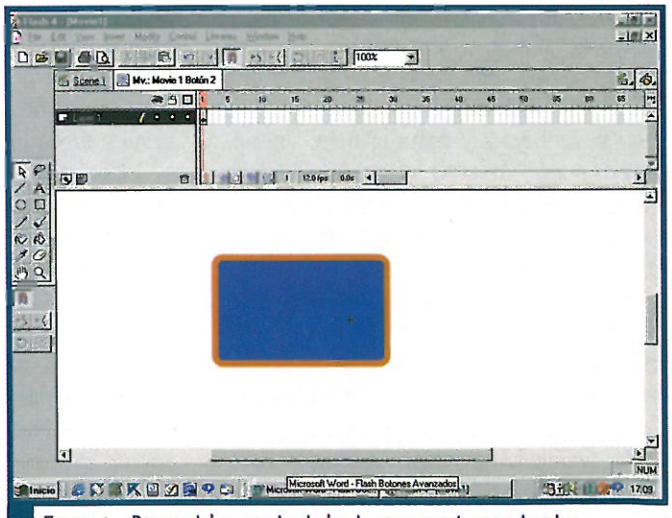


Figura 4. - Diseño del *movie* donde los botones serán introducidos.

Dreamweaver 3: Iniciación al diseño de páginas Web

Antoni Masso Mola.

Crear páginas *web* no siempre supone programar cientos de líneas sin apenas tocar el apartado de diseño. Gracias a *Macromedia DreamWeaver* 3, el arte de crear páginas *web* se libera de la programación en *HTML*. En este artículo mostraremos cómo aprender a manejar esta potente herramienta.

PROPIEDADES DE DREAMWEAVER 3

Toda persona que se haya interesado por la creación de *web sites*, ha pasado por el mismo camino. El primer "susto" lo recibimos cuando en las revistas, *tutoriales*, etcétera, nos muestran un sinfín de parámetros que en un principio nadie entiende, que es el corazón del *web*, el código *HTML*. Una vez vemos "lo complicado" que puede llegar a ser crear una *web* en *HTML* nos llega la decepción. Pero si se busca y estudia más, encontraremos que existen programas que se encargan de crear el código *HTML*, y donde nuestro único trabajo es crear el diseño de la *web* como mejor nos guste.

Uno de los más conocidos es el *Macromedia DreamWeaver* 3. Se trata de un programa multimedia con que se puede conseguir crear todo tipo de páginas *web* sin saber programar en *HTML*. Este tipo de programas están diseñados para escribir automáticamente el código tal y como el usuario desea que se muestre en un navegador.

Antes de empezar a utilizar el programa, es necesario conocer sus funciones y familiarizarse con sus botones. Una vez tengamos abierto el programa, nos aparecerá su pantalla de trabajo. Está compuesta de un gran espacio en blanco central, en la parte superior una serie de opciones, y finalmente, en el lateral derecho inferior, los botones más importantes del programa que serán estudiados más adelante. Podemos ver el aspecto del programa en su primera sesión en la Figura 1.

Como en todo programa, primero nos tenemos que familiarizar con los opciones integradas en el programa localizadas en la parte superior del mismo. Se pueden apreciar diez carpetas, donde en cada una de ellas aparece un menú despegable con las propiedades de uso similar para la creación o edición del *web*. Para poder ver mejor lo que intentamos aprender, es una buena idea acceder al programa y presionar sobre la carpeta *Insert* haciendo un clic sobre ella. El resultado debería ser el mismo que el de la Figura 2.

La primera carpeta que analizaremos será la llamada *File*. Cuando el programa es abierto como una nueva sesión, esta carpeta dispone de 16 opciones utilizables o presentes. Las cuatro primeras

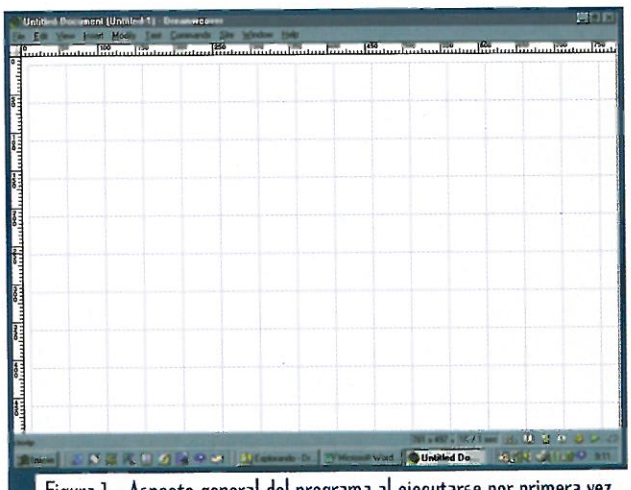


Figura 1. - Aspecto general del programa al ejecutarse por primera vez.

son muy conocidas, debido a que todo programa dispone de ellas. La primera es *New*. Presionando sobre esta opción, dispondremos de una nueva sesión con *DreamWeaver*. El programa abre una copia del mismo, en vez de crear una nueva plantilla sobre la ya utilizada. Esto es de gran ayuda para no liarse entre una o otra página *web*. Es mucho más sencillo y "limpio" poder pasar de un programa a otro, con mucha más rapidez, que tener que disminuir o aumentar una ventana dentro del programa cada vez que se desee cambiar de plantilla o trabajo. En versiones anteriores, la opción de abrir una nueva ventana como programa, no estaba disponible. Apareció por primera vez en la versión *2*.

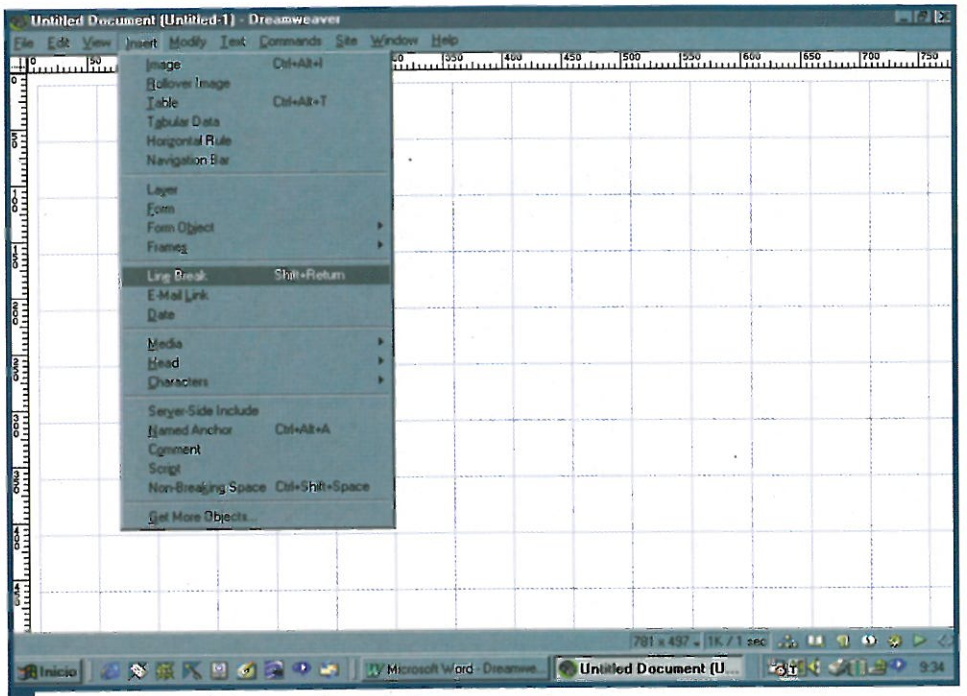


Figura 2. - Captura de la carpeta Insert, donde aparecen sus herramientas.

La primera vez que usemos DreamWeaver será sin plantillas

A continuación, nos encontramos con una interesante opción, que a medida que dominemos el programa con más detalle, será de gran ayuda. Se trata de la opción *New From Template*. Como es la primera vez que utilizamos DreamWeaver, aún no dispones de *Templates* (Plantillas), por lo que nos será imposible utilizarlo desde un principio. De todas formas, es muy importante saber la función que desempeña. Una plantilla es un documento *web* que utiliza un mismo estilo para crear otros de manera más rápida y sencilla. Una plantilla puede disponer de secciones editables y otras no editables. Por ejemplo, una *web* para la venta de productos, mantendrá el mismo estilo en toda la página. Creando una sola página *web*, es posible guardarla como una plantilla y poder utilizarla para todas las demás.

Ejecutando la opción *Open* se verán todos los documentos que pueden ser abiertos por *DreamWeaver*. *Close* es una opción igual de conocida que *Open*, que no necesita más explicación.

Las cuatro opciones siguientes están programadas para guardar el documento en las diferentes extensiones que el programa reconoce. La primera de ellas, *Save*, guarda los cambios realizados en el documento en uso. *Save As* realiza la misma operación, pero deja la posibilidad de introducirle un nombre al documento *web*. También tiene la opción de guardar el documento con una extensión diferente a *HTML*. Por ejemplo, si se está trabajando en una *web* programada en *XML*, el programa permite guardarlo bajo tal extensión.

Las plantillas son de gran importancia para los diseñadores de *webs*. Por eso, una vez hecha una *web*, es posible guardarla como una plantilla para poder utilizarla y editarla en el futuro. Esto lo podemos conseguir guardando la *web* con la opción *Save as Template*. Si intentamos guardar un posible documento *web* como una plantilla, nos aparecerán una serie de preguntas que deberemos rellenar para que sea utilizable correctamente en un futuro. Podemos ver la apariencia de tales pasos en la Figura 3.

Por último, nos encontramos con *Save All*. Esta opción nos permite guardar el documento en uso, realiza la misma operación que el *Save As*. Como hemos podido apreciar, *DreamWeaver* dispone de un número importante de opciones para guardar documentos. En este instante nos puede parecer una simple operación, pero en un futuro, averiguaremos cómo *Save As Template* nos puede ser de gran ayuda.

De las siguientes opciones, sólo estudiaremos las más importantes. La primera de ellas es *Import*. Puede importar otro tipo de documentos, como *Word* o *XML* e insertarlas en la *web*. La siguiente es muy similar a *Import*, realiza la misma tarea pero exportando el documento. Es llamada *Export* y puede exportar la web como un documento XML, CSS o un tabla.

Debido a que no todos los usuarios utilizan las últimas versiones de los navegadores, existe la posibilidad de hacer la *web* compatible con los antiguos navegadores 3.0. Si esta opción no estuviera presente, aquellos usuarios con un navegador 3.0 no visualizarían la *web* con exactitud.

Podemos hacer la web compatible con los antiguos navegadores 3.0

Por último, nos encontramos con *Preview in Browser*. Traducido al castellano, *Vista Previa en el Navegador*. Con esta opción podemos ver cómo está quedando el diseño de la *web* en cada momento y con el navegador que deseemos.

A parte de *File*, existen otras dos carpetas de igual importancia, *Insert* y *Modify*, que son el motor del programa. Con ellas podemos crear y editar el diseño de la *web*. Las otras carpetas no depositan opciones con tanta importancia, por lo que las dejaremos para próximos artículos

La carpeta *Insert* es donde residen las opciones de creación del *web*. En ella se encuentra la posibilidad de introducir imágenes, tablas, texto, sonido, multi-

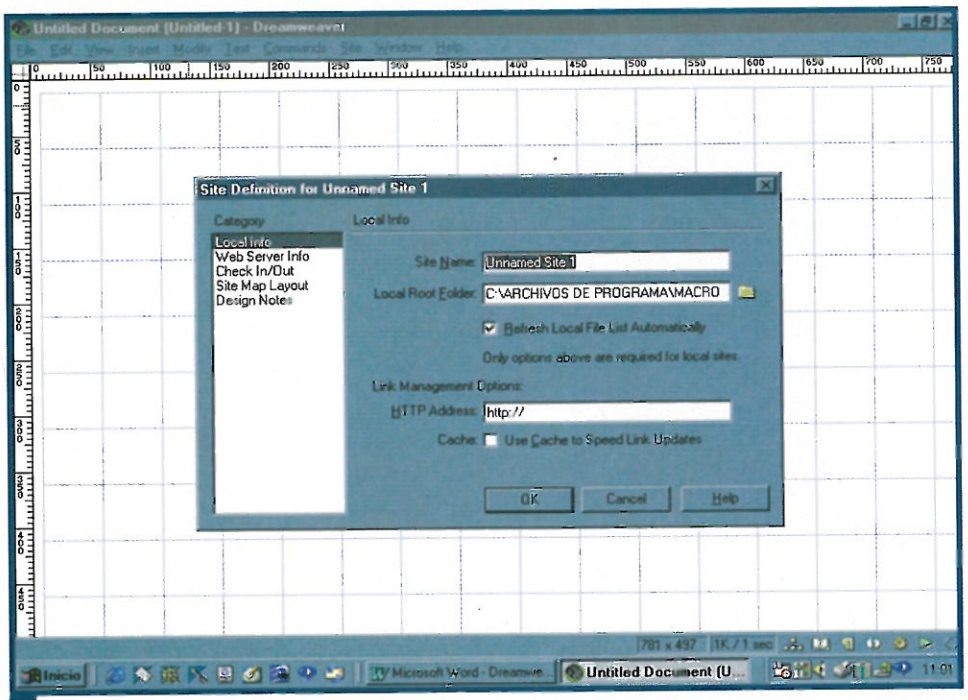


Figura 3. - La herramienta *Save as Template* en funcionamiento.

media, etcétera. Por eso es muy importante que nos concentremos principalmente en esta carpeta.

La primera opción con la que nos encontramos es *Image*. *Image* se refiere a la opción de introducir un gráfico. Existen dos tipos de extensiones utilizables en el *web*: *.gif* y *.jpg*. La primera de ellas, es depeor celidad que la segunda, y ocupa menos memoria. Cuando se desee emplear una u otra extensión, es recomendable utilizar la *.gif*, dado que facilitará la descarga del *web*.

Un método de organizar una web es introduciendo una tabla

Haciendo clic sobre sobre *Image*, una pantalla flotante aparecerá, invitándonos a seleccionar el gráfico deseado, ya sea dentro de nuestro disco duro o en una red. Una vez tengamos el archivo seleccionado, presionando sobre *Select* para que aparezca en pantalla. *DreamWeaver* automáticamente lo sitúa al lado izquierdo del programa. En próximos artículos, se mostrará el método para situar un gráfico en un lugar exacto.

La siguiente opción dentro de la carpeta *Insert* es *Rollover Image*. Es una interesante opción con la cual se puede conseguir llamar la atención de los usuarios de un manera clara y concisa. Lo que se consigue es dar el efecto de cambio de imagen. Por ejemplo, disponemos de un botón rojo y deseamos que éste pase a verde cuando el *mouse* se sitúe sobre él. Si tuviéramos que programarlo, en estos momentos nos sería casi imposible. Pero con *DreamWeaver* no es necesario saber cómo programar, nuestro único trabajo es introducir toda nuestra creatividad. Para el cambio de color, sólo debemos crear dos botones idénticos, pero uno con el color rojo y el otro verde. Con ambos botones, podemos introducirnos en la opción *Rollover Image*. El programa nos preguntará el nombre desea, éste puede ser cualquiera, por ejemplo, *botón 1*, para así saber que es el primero de entre todos. La siguiente casilla, *Original Image*, hace referencia al primer gráfico. *Rollover Image* es el gráfico secundario, que será utilizado cuando el ratón se sitúe sobre él. Por último nos encontramos con una casilla que sirve para indicar el *URL* o página *web* donde queremos que el botón lleve a nuestro visitante. Una vez introducidos los datos, presionando sobre *OK*, el programa crea el código necesario y automáticamente disponemos de un botón animado en nuestra primera página *web*.

Un método de organizar una *web* es introduciendo una tabla. Gracias a las tablas es posible crear *webs* claras con los bordes rectos. Realizar todo esto es muy sencillo. Accediendo a *Table*, dentro de la carpeta *Insert*, podremos disponer de una tabla en cuestión de segundos. En la pantalla de creación de tablas es posible seleccionar el número de líneas horizontales y verticales, la separación entre ellas, el tamaño y el grosor de las mismas...

Hay otras opciones que no trataremos, debido a su nivel de complicación y a que sin haber practicado antes con las anteriores herramientas, es muy difícil averiguar cómo manejar un gran número de las que preceden. Nos centraremos en las más sencillas, pero igual de importantes que el resto.

Nos situaremos en la opción *E-mail Link*. Únicamente es necesario introducir el texto deseado que hará la función de link y la

dirección de correo electrónico.

Para aquellos que deseen introducir sus creaciones de *Flash* u otro programa externo, pueden hacerlo con la opción *Media*. Seleccionando el tipo de archivo que deseamos introducir y escogiéndolo dentro de la nueva ventana, dispondremos del archivo externo deseado.

Tras haber analizado por encima las opciones más importantes, nos es posible empezar a crear una *web*. Pero pronto nos surgirán problemas si no sabemos modificar los objetos introducidos. Para eso ello, existe una carpeta llamada *Modify*. Dentro de ella se encuentran las opciones más interesantes para poder editar todo lo creado.

Page Properties reúne todas las variables de la página principal

Una de las opciones de edición más importantes es la llamada *PageProperties*. En ella se encuentran. La primera opción es *Title*, lugar donde introduciremos el título de la *web*. Seguidamente aparece una casilla que nos presenta el medio de introducir una imagen (*.gif* o *.jpg*) para ser utilizado como fondo de la *web*. Luego nos indica los diferentes colores que se pueden editar para el fondo, el texto, los *links*

(visitados y activados). Además es posible elegir los márgenes dela *web*, el tipo de fuente y la transparencia de las imágenes en ella expuestas.

La próxima opción es la *Selection Properties*. Esta, al igual que la anterior,

dispone de opciones de maquetación del *web*. Si ningún objeto es seleccionado, las propiedades introducidas afectarán únicamente al texto. Se puede alterar el tamaño, formato, fuente, situación y color del mismo. Si se desea actuar sobre otro objeto, tan sólo deberemos seleccionar el que queramos modificar e introducir los cambios, por ejemplo, el de centrar una imagen.

La siguiente función es un poco más avanzada. Con ella necesitamos conocimientos de programación *HTML*. Su utilidad es la de introducir una sentencia en el lugar deseado. Por ejemplo, si se desea introducir un texto en negrita, ejecutaríamos el programa y teclearíamos: * Texto en negrita *. De este modo nos ahorramos tener que ver el código fuente de la *web*, encontrar, entre toda la programación, el lugar exacto donde introducir esa pequeña sentencia. Es una buena ayuda para aquellas personas que dispongan un mínimo conocimiento de HTML.

La próxima opción de nivel básico es la *Layout Mode*. Con esta herramienta, cambiar una capa por una tabla o viceversa no supone ningún problema, ya que tan sólo debemos seleccionar aquella capa o tabla deseada dejando el resto del trabajo al programa.

LA BARRA DE HERRAMIENTAS LAUNCHER

Centrémonos ahora en averiguar para qué sirve la utilidad de la barra de herramientas *Launcher*. Para activar tal barra, nos deberemos introducir en la carpeta de *Window* y hacer clic sobre *Launcher*.

La barra de herramientas flotante *Launcher*, está compuesta de siete opciones. La primera de ellas, *Site*, sirve para administrar todas las *webs* incluidas en una misma página *web*. Por ejemplo, es de uso común para almacenar *webs* de gran envergadura. En vez de guardar todos los ficheros *HTML* en un mismo directorio, por lo que podría complicar a más de a uno la tarea de subirlas a *La Red*, ya sea mediante *ftp* o el *web*. Una posibilidad es conectarse con el mismo programa a una posible cuenta de dominio en un servidor, para que el usuario no deba hacer nada, todo sería automatizado. Para llegar a conocer y utilizar esta herramienta, es necesario practicar con otras de mayor sencillez.

La barra de herramientas flotante Launcher, está compuesta de siete opciones

El próximo botón de la barra de herramientas es *Library*. Es exactamente lo que el nombre indica, una librería. Dentro de una librería aparecen aquellos objetos que se usan dentro de la *web* con regularidad. Éstos pueden ser imágenes, texto o cualquier otro elemento. Cuando se inserta un elemento de la librería dentro del documento, *DreamWeaver* automáticamente crea el código *HTML* de ese objeto y lo referencia a la librería. De este modo, cambiar la función de un botón que es utilizado a lo largo de una *web site* entera, puede hacerse cambiando el código *HTML* de la imagen en la librería. Gracias a esta herramienta, no es necesario cambiar la función para cada página, ya que el programa lo hace automáticamente.

Los siguientes dos botones no los usaremos hasta más adelante, debido a su dificultad y poca utilidad en estos momentos. Se trata de las herramientas *HTML Styles*, *CSS Styles* y *Behaviors*. Su finalidad es crear textos y caracteres que puedan ser guardados en una clase de librería para utilizarlos cuando sea necesario.

El botón *History* es muy interesante. Gracias a él, podremos averiguar cada una de las acciones que hemos realizado en el documento desde que hemos empezado a trabajar.

El programa captura todo tipo de movimiento dentro del documento y lo almacena en un tipo de base de datos, donde podemos acceder y visualizar todos los pasos realizados. Es una importante herramienta para analizar el lugar donde se cometió un error y poder solucionarlo. Debido a que guarda todos los pasos realizados, si un usuario no recuerda cómo consiguió hacer que un objeto pasara de un lado al centro, *Behaviors* se lo mostrará.

Por último, sólo nos falta analizar el botón *HTML Source*, quizás es el más importante de todos. Dado que es un programa multimedia, la programación no es necesaria en casi ningún aspecto, pero sí lo es cuando una web es más compleja de lo normal. Esta herramienta muestra el código de una forma muy clara. Separa las diferentes funciones entre colores, y el texto lo mantiene de un mismo color. Así es muy sencillo resolver cualquier problema de programación. También es una buena manera para que aquellos que no disponen de conocimientos de programación en *HTML*, puedan aprender un gran número de funciones, y así, familiarizarse con él. Si un objeto o texto es seleccionado en el documento, es posible abrir el editor de *HTML*, para ver cuál es el código utilizado.

CONCLUSIÓN

En este primer contacto con el programa *DreamWeaver* 3 de *Macromedia*, hemos podido averiguar sus funciones más importantes. A partir de ahora, estaremos preparados para empezar a trabajar y aprender cómo funciona el programa. Podremos crear nuestra primera página *web* que utilice imágenes, texto, botones, propiedades de fondo, etcétera.

En el próximo artículo nos centraremos en cómo hacer que las imágenes, botones y texto mantengan un concordancia entre sí, además de explorar a fondo las funciones de maquetación y diseño.

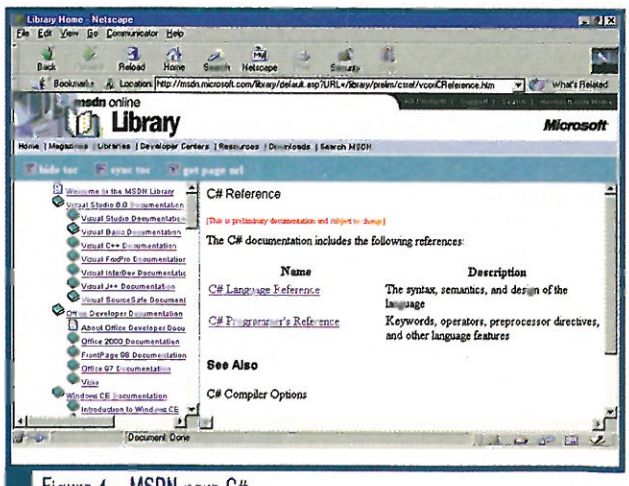
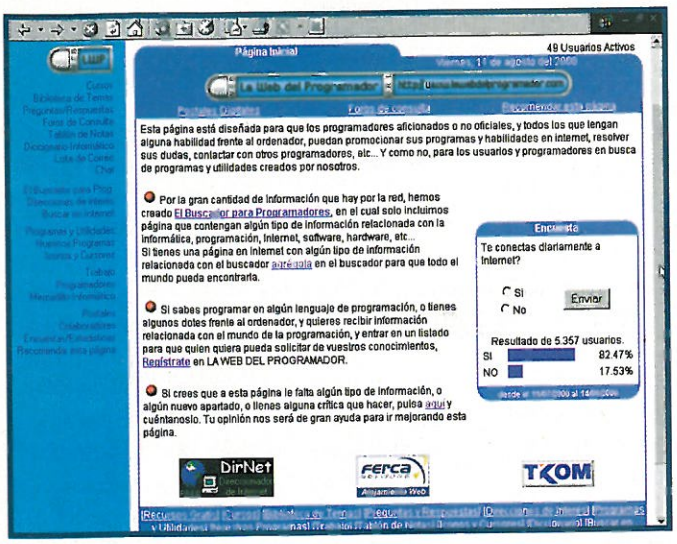


Figura 4. - MSDN para C#.

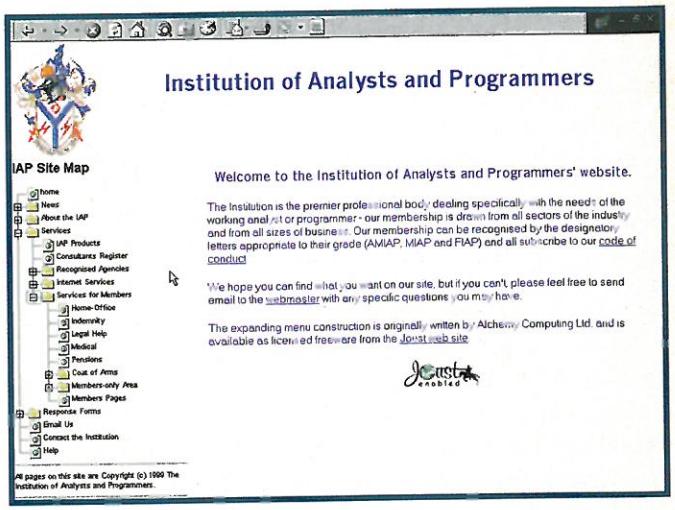
Direcciones de Interés

www.lawebdelprogramador.com


Página Inicial
49 Usuarios Activos
Esta página está diseñada para que los programadores aficionados o no oficiales, y todos los que tengan alguna habilidad frente al ordenador, puedan promocionar sus programas y habilidades en internet, resolver sus dudas, contactar con otros programadores, etc... Y como no, para los usuarios y programadores en busca de programas y utilidades creados por nosotros.
Por la gran cantidad de información que hay por la red, hemos creado El Buscador para Programadores, en el cual solo incluimos página que contengan algún tipo de información relacionada con la informática, programación, internet, software, hardware, etc...
Si tienes una página en internet con algún tipo de información relacionada con el buscador apúntala en el buscador para que todo el mundo pueda encontrarla.
Si sabes programar en algún lenguaje de programación, o tienes algunos dotes frente al ordenador, y quieres recibir información relacionada con el mundo de la programación, y entrar en un listado para que quien quiera pueda solicitar de vuestros conocimientos, Regístrate en LA WEB DEL PROGRAMADOR.
Si crees que a esta página le falta algún tipo de información, o algún nuevo apartado, o tienes alguna crítica que hacer, pulsa aquí y cuéntanoslo. Tu opinión nos será de gran ayuda para ir mejorando esta página.
Encuesta
Te conectas diariamente a Internet?
SI
No
Enviar
Resultado de 5.357 usuarios.
SI 82.47%
NO 17.53%
DirNet
FERCA
TKOM

Interesante página,con buscadores y foros de consulta. En ella encontraréis, desde un mercadillo hasta una biblioteca de temática. Se actualiza diariamente.

http://www.iap.org.uk

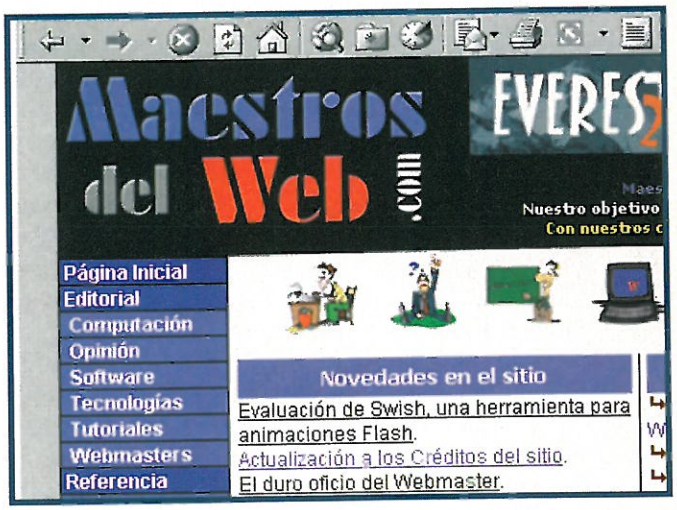

Institution of Analysts and Programmers
IAP Site Map
Welcome to the Institution of Analysts and Programmers' website.
The Institution is the premier professional body dealing specifically with the needs of the working analyst or programmer - our membership is drawn from all sectors of the industry and from all sizes of business. Our membership can be recognised by the designatory letters appropriate to their grade (AMIAP, MIAP and FIAP) and all subscribe to our code of conduct
We hope you can find what you want on our site, but if you can't, please feel free to send email to the webmaster with any specific questions you may have.
The expanding menu construction is originally written by Alchemy Computing Ltd. and is available as licensed freeware from the Joust web site
All pages on this site are Copyright (c) 1999 The Institution of Analysts and Programmers.

Pagina de la Institución de Analistas y Programadores. Muy británica toda ella. Para ser miembro de la IAP hay que alcanzar una puntuación de acuerdo a un baremo al efecto, y jurar un código de conducta.

http://www.maestrosdelweb.com

Página de Bass Software, un grupo de programadores españoles locos por los videojuegos. Puedes presentar tus proyectos y conocer los de otros programadores.

http://www.maestrosdelweb.com



Completa página dedicada al webmaster con manuales, foros, enlaces, programas, buscadores... será difícil no encontrar aquí lo que buscas.

LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SÓLO PROGRAMADORES

solop@revistasprofesionales.com

El Poder es Tuyo. ¡Elige Ahora!

Special Program Available for Resellers

Disponible en:

abc analog, s.l.
✆ 91 634 20 00

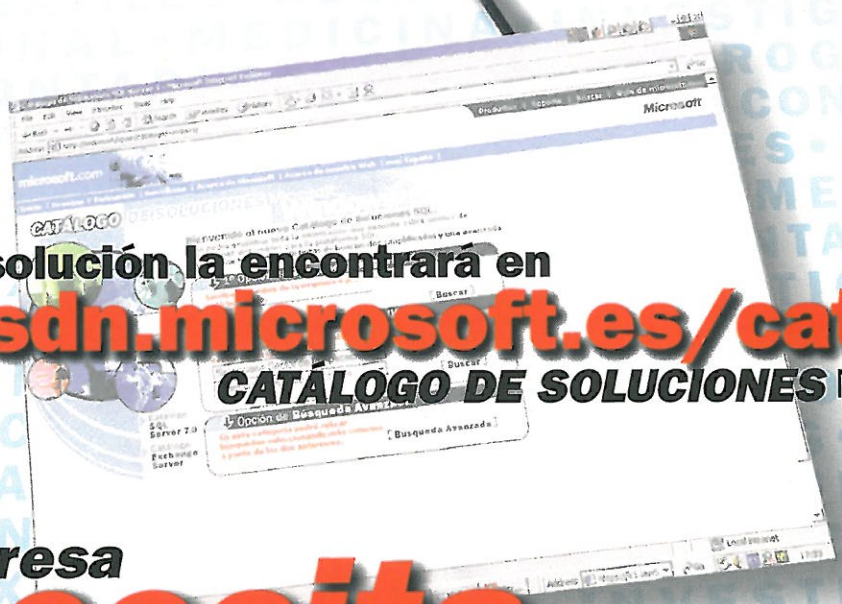
GTI
✆ 91 660 08 30

redhat.
www.redhat.com
Red Hat (ESPAÑA) S.L.

Su empresa
Tiene

Aplicaciones desarrolladas bajo Microsoft Windows

"Nuestra empresa ha desarrollado una aplicación. ¿Cuál será el sitio donde la puedan conocer el mayor número de clientes potenciales?"



La solución la encontrará en

msdn.microsoft.es/catalogo

CATÁLOGO DE SOLUCIONES Microsoft **Windows**

El lugar donde se encuentra el mayor número de aplicaciones desarrolladas bajo Microsoft Windows para cualquier tipo de empresas.

Su empresa
necesita

Aplicaciones de desarrollo bajo Microsoft Windows

"Buscamos una aplicación de gestión a medida para la actividad de nuestra empresa"

Las 50 primeras soluciones* que se incluyan en nuestro catálogo de soluciones, obtendrán un ratón IntelliMouse Explorer o un teclado Microsoft Internet Keyboard

* Solamente se obtendrá un regalo por empresa

Microsoft